Phishing in Wonderland: Evaluating Learning-Based Ethereum Phishing Transaction Detection and Pitfalls

Ahod Alghuried [1,2] (✉) and David Mohaisen[1,3] (✉)
[1]*University of Central Florida* [2]*Prince Sattam bin Abdulaziz University* [3]*Ewha Womans University*

***Abstract*—Phishing attacks pose significant risks to the Ethereum ecosystem, comprising over 50% of Ethereum-related cybercrimes, leading to the emergence of many machine learning-based defenses. This paper introduces a comprehensive framework aimed at enhancing machine learning-based phishing detection in Ethereum transactions. The framework addresses critical aspects such as feature selection, class imbalance, model robustness, and algorithm optimization. By systematically evaluating the strengths and limitations of existing approaches, we highlight gaps in current practices, particularly in feature manipulation and unsustainable performance outcomes. Through both analytical and experimental assessments, we demonstrate the framework's ability to streamline detection techniques, improving generalization and model effectiveness. Our findings emphasize the importance of refining detection strategies to meet the evolving challenges posed by sophisticated phishing schemes in the blockchain space.**

I. Introduction

Among the many security concerns related to Ethereum [1], [2], [3], [4], [5], [6], phishing scams stand out [7], [8], [9], [10], constituting over 50% of Ethereum-related cybercrimes since 2017 and emerging as a significant threat to the trading security of Ethereum [11]. Unlike traditional phishing, which relies on deceptive emails or fake websites to steal sensitive information, Ethereum phishing exploits the blockchain's transparency by using fraudulent smart contracts or addresses that appear legitimate to lure victims. Ethereum transactions are the core operations on the blockchain, facilitating asset transfers, smart contract executions, and account interactions [12], [13]. Each transaction records sender and receiver addresses, transaction value, gas fees, and timestamps, making every action verifiable on the blockchain ledger [14], [15].

Ethereum's support for smart contracts–encoded agreements that automate complex interactions–enhances its versatility but also introduces vulnerabilities. Attackers can create deceptive contracts or manipulate transaction metadata to mimic legitimate interactions, taking advantage of the blockchain's pseudonymous and transparent nature [13], [16]. This transparency, while integral to blockchain, complicates phishing detection, as malicious transactions can blend seamlessly into normal activity [16], [17].

Ethereum's susceptibility to phishing is heightened by the high-value transactions, pseudonymity, and the open visibility of transaction data, which enable attackers to design targeted, deceptive scams [18], [15]. In contrast to traditional phishing, where personal information is typically the target, Ethereum phishing leverages the programmable nature of the blockchain to alter transaction records and smart contracts, making attacks difficult to detect. These features make Ethereum particularly attractive to phishing schemes [19], [20]. The Bee token Initial Coin Offering (ICO) scam of 2018 resulted in a loss of one million USD within 25 hours [21]. The phishing attack on Uniswap Labs users in 2022 caused losses exceeding eight million USD. In 2024, reports alleged the discovery of 91 wallets that amassed more than two billion USD from illicit activities, including scams, on Ethereum [22]. All these incidents highlight the urgent need for stronger measures to protect both assets and user trust in the cryptocurrency [23].

Phishing scams have evolved to exploit cryptocurrencies' unique intricacies, posing significant security threats to their integrity [24]. Financial losses from successful attacks also undermine confidence in blockchain technology, and the persistent threat of such cybercrimes casts a shadow over the trust in digital currencies [21], [25], [13]. This evolving threat has prompted a surge in research efforts to fortify cryptocurrencies against these sophisticated cybercrimes.

Numerous research works combined machine learning with cryptocurrency analyses to combat phishing attacks. Since transactions form a graph, many of these studies detect phishing employing graph-based analysis [26], [9], [27]. Analyzing blockchain activity alone aims to develop machine learning methods to identify malicious activities. For example, Chen *et al.* [26] employed convolutional networks and auto-encoders alongside transaction graphs to identify phishing accounts. Similarly, Lou *et al.* [28] used a Convolutional Neural Network (CNN) to improve the precision and recall of detection. Wu *et al.* [15] implemented a one-class learning for deciphering structural relationships in Ethereum's transaction network using Graph Convolutional Networks (GCN) and autoencoders for high precision and recall for transaction classification.

Nevertheless, these works fail to systematically understand the circumstances under which such approaches deliver outstanding results. In particular, proper implementation and analysis of those systems are lacking as they often make ad hoc assumptions about how phishing works in reality in a way that appeases the implementation of the machine algorithms to deliver high effectiveness. Such assumptions may not necessarily be grounded in the reality of the cryptocurrency.

This work examines prevailing practices in machine learning-based phishing detection for Ethereum, with a focus on system operation, dataset composition, feature selection, robustness evaluation, and model optimization. Motivated by

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240694
www.ndss-symposium.org

the urgent need to strengthen phishing defenses [29], [30], we systematically categorize and assess detection features, algorithms, and techniques, identifying both strengths and critical shortcomings. Our analysis reveals substantial disarray in the field. Feature selection is frequently inconsistent, and dataset composition is often manipulated to artificially balance class labels and inflate performance metrics. Many proposed features are vulnerable to manipulation, while robustness evaluations are frequently neglected, raising concerns about the sustainability of reported performance. Additionally, models tend to be over-parameterized without justification, although more streamlined architectures using fewer features can achieve comparable results.

Contributions. In this paper, we make the following contributions. (1) We build a systematic evaluation framework that unveils the underlying practices in learning-based phishing detection in Ethereum, including the common practices with feature selection, dataset composition, feature robustness examination, and feature selection and model optimization. We analyze and categorize a comprehensive list of recent studies in this space and highlight where they fail to implement good and sustainable practices. (2) We empirically assess several studies from the literature using our analytical framework, addressing various research questions by quantifying concerns highlighted within the framework. Additionally, we examine practices surrounding these studies, highlighting the risks of overlooking fundamental tests and assessments.

For consistency, we use *phishing transactions* as the primary term throughout the paper, and we treat it as encompassing transactions initiated by phishing accounts as well as transaction-level manifestations of broader phishing activities.

Scope Clarification.. Directions commonly explored in operational phishing-detection systems, such as temporal embargoes, campaign-disjoint data splits, live evasion simulation, and continuous benchmarking, are not applicable to our setting. The Ethereum dataset used in this study is static and retrospective, with all transactions historically finalized. Under these conditions, temporal unfolding, adversarial interaction, and red-team style evaluation cannot be meaningfully incorporated. Our evaluation design instead reflects the retrospective analysis conditions of prior published studies, preserving transaction-level independence to avoid leakage while ensuring comparability with the pipelines we assess.

Organization. In section II we presented the preliminary work, followed by our research questions and pipeline in section III, analytical framework in section IV, experimental evaluation in section V, and concluding remarks in section VI.

II. Preliminary Work

A. Background on Phishing Detection

Ethereum security has been a hot research topic [31], [32], and many studies have been dedicated to understanding and detecting phishing on the Ethereum network [20], [33]. Early comprehensive studies have shed light on Ethereum-based Ponzi schemes, highlighting the tactics adversaries employ. Moreover, subsequent research efforts have delved deeper, uncovering concealed Ponzi schemes that exploit smart contracts,

TABLE I: The literature on phishing accounts detection using various features, features groups, and algorithms across various evaluation metrics.

Work	Group	Features	Algorithms	Performance	
				F1	AUC
Chen *et al.* [21]	Transaction	T, A	LightGBM	0.80	0.81
Chen *et al.* [26]	Transaction	ID, OD, D, IS, OS, S, N	LightGBM	0.16	0.56
Wen *et al.* [38]	Transaction	T, A, ID, OD, N, from, to	SVM, KNN, AdaBoost	0.94	0.92
Kabla *et al.* [25]	Transaction	T, BN, from, input	KNN, DT	0.97	0.97
Li *et al.* [40]	Transaction	ID, OD, D, A, T, NT	LightGBM	0.81	0.92
Wu *et al.* [15]	Transaction	T, A	SVM	0.90	–
Palaio. *et al.* [37]	Transaction	BN, GF, NT, ST	SVM, CNN, XGBoost	0.85	–
Li *et al.* [9]	Transaction	ID, OD, D, A, T, NT	XGBoost	0.92	–
Lin *et al.* [7]	Transaction	BN, A, GL, GF	Neural network	0.82	–
Wen *et al.* [41]	Transaction	T, A, TD, GF, B, NT	Neural Network	0.97	–
Zhou *et al.* [42]	Transaction	T, A, GF, ID, OD	EGAT	0.97	–
Chen *et al.* [26]	Behavioral	ID, OD, D, IS, OS, S, N	LightGBM	0.16	0.56
Wen *et al.* [38]	Behavioral	ID, OD, N	SVM, KNN, AdaBoost	0.94	0.92
Li *et al.* [40]	Behavioral	ID, OD, D	TTAGN	0.81	0.92
Li *et al.* [9]	Behavioral	ID, OD, D	TGC	0.92	–
Zhou *et al.* [42]	Behavioral	ID, OD	EGAT	0.97	–
Dong *et al.* [43]	Content	ID, OD, NT	SVM, LR	–	0.97
Wen *et al.* [41]	Content	T, A, TD, GF, B, NT	CNN	0.97	–

(1) Features: Time (T), amount (A), in-degree (ID), out-degree (OD), degree (D), in-strength (IS), out-strength (OS), strength (S), neighbors (N), block number (BN), gas fee (GF), number of transactions (NT), successful transactions (ST), gas limit (GL), balance (B), transfer direction (TD). (2) Metrics: F1 score and area under the curve (AUC).

leveraging advanced data mining and machine learning methods to evade detection [34], [15], [7]. These studies highlight the evolving nature of scams and catalyze the development of more robust detection and prevention mechanisms.

Phishing scams typically deceive individuals by impersonating trusted entities to obtain sensitive details, including usernames, passwords, and financial information [35]. These scams often involve fraudulent communication, such as emails, masquerading as legitimate, attempting to gather personal information from victims. Phishing operations are known for their ephemeral nature, emerging suddenly to capture victims and dissipating just as quickly. More personalized and insidious variations, known as spear-phishing, target a narrower but potentially more lucrative demographic [36].

Phishing Detection. In response to this threat, various works have been proposed, initially focusing on transactional features, moving to behaviors and contents:

Transactional Features. Prior work has explored time, amount, degree-based, and block-level metrics in a variety of ML pipelines [15], [7], [26]. Details are provided in Appendix B1.

Behavioral Features. Prior work has examined behavioral indicators such as transaction success rates, DeFi interaction patterns, and account-level behavioral signals for phishing detection [37], [38]. Details are provided in Appendix B2.

Content Features. Content-based approaches leverage transaction-level contextual data, including network-graph structures, edge attributes, and domain or website-derived information [24], [39]. Details are provided in Appendix B3.

B. Comparative Literature Analysis

Table I provides an overview of representative studies across transactional, behavioral, and content feature groups. It summarizes commonly used features, algorithms, and reported performance metrics, establishing the context for our feature-group taxonomy and the evaluation practices revisited in later sections. Additional descriptive discussion of individual studies is provided in Appendix B4.

The table also presents evolving strategies for detecting phishing, delineated across transactional, behavioral, and content dimensions. Notable studies by Kabla *et al.* [25] and Zhou *et al.* [42] achieve high efficacy (F1 and AUC of 0.97), showing the effectiveness of well-tailored feature sets. In contrast, lower performance in studies such as Chen *et al.* [26] (F1 of 0.16, AUC of 0.56) is shown. The comparison of these studies highlights the diversity of features from primary transaction data to intricate network behaviors.

Expanding Detection. Our study seeks to explore and test the robustness of these feature groups in phishing detection. We aim to replicate such works Kabla *et al.* [25] and Chen *et al.* [26], delving into the impacts of various sets on the efficacy of learning models in detecting phishing activities.

III. Questions and Methodology

A. Research Questions

This work focuses on five questions to understand the robustness, generalization, reproducibility, and comparison of Ethereum phishing detection, which we highlight below.

RQ1. **Are the prior works in the literature on phishing detection doing the proper feature selection?** This question stems from the observation that the features designed for detection in the prior work [26], [38], [42], [21], [9], [40], [41], [43] lack thorough consideration and justification. Frequently, these features are devised hastily and lack proper statistical analysis. In other cases, those features result from a black-box learning module that is difficult to comprehend.

RQ2. **What is the impact of the representation of phishing as benign on the performance of the phishing detection algorithms?** This question stems from the observation that, in particular works, an arbitrary number of phishing transactions is assumed in the datasets [21], [25], [21], [37], [40], [9], [7], [41]. To achieve a balanced ratio between benign and phishing samples, these studies frequently introduce random phishing transactions into their datasets without considering the impact on other features utilized in the classification process, regardless of whether they are dependent or independent.

RQ3. **How do different algorithms compare to one another?** This question arises from observing that various approaches [15], [7], [37], [41], [26], [38], [24], [21], [43] employing distinct features and feature groups, and different algorithms, are typically conducted independently and not compared against each other in terms of their performance using comparable evaluation metrics and consistent evaluation settings. An additional aspect of answering this question involves determining the reproducibility of the findings from previous studies on cryptocurrency phishing detection.

RQ4. **What impact does dataset preprocessing have on the effectiveness and generalization of detection?** Many studies incorporate a preprocessing phase, e.g., reducing a larger network to a smaller one to make the analysis and detection algorithm implementation more feasible computationally [9], [40], [15], [7], [26]. However, such preprocessing excludes certain underlying transaction networks while possibly exaggerating others that may not be as prominent in the original network. Therefore, comprehending the influence of preprocessing steps on the outcome of the detection algorithm is essential.

RQ5. **How do different features and feature groups compare for robustness to manipulation?** An effective and sustainable detection algorithm should depend on robust features not easily prone to manipulation by adversaries to evade

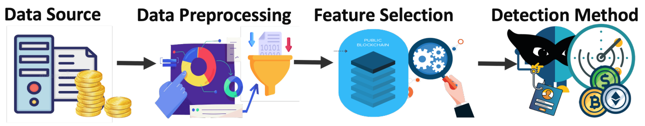


Fig. 1: Detection pipeline in cryptocurrency transactions.

detection. However, feature sets in these detection schemes (e.g., [15], [7], [42], [41], [37], [9]) are non-uniform and differ in this aspect. Conducting a systematic and comprehensive analysis of these features is crucial.

B. Methodology

Per Fig. 1, our methodology (pipeline) comprises the following steps: data collection, data preprocessing, feature selection, and detection. We will review our pipeline to address the research questions below. We limit ourselves in this discussion to the high-level and general techniques description since the individual techniques are studied in the subsequent sections when discussing specific works.

Data Source/Collection. Our pipeline begins by gathering data associated with the transactions from a data source. Such data includes full transaction information confirmed in blocks and cryptocurrency-specific application features and details. For instance, this encompasses block numbers, time, sender and receiver addresses, transaction values, and gas fees. This comprehensive dataset forms the foundation for our subsequent preprocessing and analysis steps, facilitating a thorough examination of the transactional ecosystem to identify and classify phishing attempts effectively.

Data Preprocessing. Different approaches utilize different preprocessing techniques, which we will discuss further.

Normalization. This involves converting transactions into a usable format for classification, e.g., hexadecimal to integers and extracting transaction relationships, e.g., a graph with addresses as nodes and transaction values as edges [26]. Normalization ensures data uniformity, which is crucial for robust model training. However, preserving anomalies significant for phishing detection is important, as over-normalization can reduce model sensitivity to outliers.

Scaling. Given the significant variation in transaction data magnitude, this step is crucial to avoid potential biases by adjusting the range of data features to a standard scale [25]. Commonly employed approaches include Min-Max scaling and Z-score normalization. Min-Max scaling adjusts the data to fit within a specific range, e.g., 0 to 1, proving beneficial when dealing with features to be bounded within a specific range. Proper scaling preserves the predictive strength of key variables while ensuring that infrequent yet critical phishing indicators retain their influence in the model.

Feature Selection. Identifying the most relevant features within a dataset is critical for developing robust detection models [44]. This involves selecting features that significantly affect the accuracy of phishing detection, which may include the amount and frequency of transactions, distinct transaction patterns, and deviations in transactional behaviors [40], [41].

Considerations might include account age, network breadth, and other behavioral indicators [26], [25].

General Selection. Feature selection involves a combination of ranking techniques and a voting strategy. Candidate features are first ranked using multiple criteria, and those consistently scoring above a predefined threshold are selected through voting. This approach helps retain only the most impactful features, improving model efficiency and generalization.

Correlation-based Selection. This approach is used to identify the most impactful features based on their pair-wise correlations: features with high correlation are excluded from the candidate set [25]. This reduces redundancy in the model, which can enhance performance on unseen data by focusing on independent predictors.

Engineered Features. Adopting a graph-based perspective, denoted as $G(V, E)$, this approach focuses on a specific subgraph $G_s(V_s, E_s)$ to analyze node attributes and their relationships. Key features include the in-degree and out-degree of transaction nodes, as well as the frequency and patterns of transactions [26], [7]. This method enables a fine-grained understanding of the network's structural dynamics, offering deeper insight into transaction behavior and potential associations with phishing activities.

Detection Methods. The detection stage identifies phishing using supervised ML models trained on labeled on-chain data. These models learn discriminative transactional patterns, temporal behavior, flow structure, and value dynamics, and use them to estimate the likelihood of malicious behavior, enabling reliable differentiation between benign and phishing activity.

Supervised Learning. These methods train algorithms on labeled datasets where outcomes are known, allowing the models to learn predictive patterns and apply them to unseen transactions. We employ classifiers such as LightGBM, Decision Trees (DT), and K-Nearest Neighbors (KNN). Additional details on these algorithms are provided in the appendix.

Evaluation and Validation. We use various metrics for evaluation: accuracy, precision, recall, and F1 score. ① *Accuracy.* The accuracy denotes the ratio of correctly predicted observations to the total observations, offering an insight into the model's differentiation between phishing and legitimate transactions. ② *Precision.* The precision is the ratio of correctly predicted positive observations to the total predicted positive observations. ③ *Recall.* The recall, or sensitivity, quantifies the proportion of correct positives correctly identified. ④ *F1 Score.* The F1 score, also known as the harmonic mean of precision and recall, verifies the model's accuracy by factoring in the false positives and false negatives. ⑤ *The Area Under the Curve (AUC)* AUC is indispensable, especially in situations characterized by class imbalance, a frequent occurrence in phishing detection. The AUC metric assesses the model's ability to differentiate between phishing and legitimate transactions, with a perfect model achieving an AUC of 1. An AUC of 0.5 suggests a performance no better than random chance.

IV. Analytical Evaluation

To answer the research questions in Section III-A, we take two complementary approaches: a theoretical analysis grounded in the operational context of phishing detection, and an experimental evaluation based on implementing representative schemes to validate that analysis. The resulting findings are presented in Section V, where we provide quantitative evidence supporting the insights developed here.

A. Feature Selection Techniques

The prior works have employed various techniques to select features for building the learning model. Those features are shown in Table I for the different techniques.

1) Feature Selection: Our primary questions are: (1) How are features selected? (2) Is there a standard for their selection? (3) Does the technique for selecting those features directly address their importance? (4) Are they all essential for the functioning of these schemes? To contextualize our discussion, we present the different feature selection techniques in the works discussed in Table I.

Trans2vec. Introduced by Wu *et al.* [15], Trans2vec stands out as a specialized network embedding technique designed for transaction networks like Ethereum. It is notable for its feature extraction method, embedding transaction-specific attributes into node representations focusing on transaction amount and time. By constructing a multidimensional feature space, Trans2vec represents each node within the transaction network, where transaction amount and timestamp are not merely additional data points but are intricately incorporated into each node's representation structure.

Cascading. The cascading method [21] begins by analyzing individual accounts based on transaction characteristics like frequency and amount, then progressively incorporates features from first-order, second-order, and higher-order connections. This hierarchical feature construction captures both individual behaviors and broader network interactions for effective phishing detection.

Correlation Analysis. Correlation analysis is essential for uncovering linear dependencies among features and identifying potential redundancy [37]. High correlation suggests interdependence, which can undermine model efficiency. To enhance performance, correlation-based techniques often remove features that exhibit strong mutual correlation.

Feature Engineering. Wen *et al.* [38] takes a targeted approach to Ethereum transaction analysis, distinguishing between Account Features (AFs) and Network Features (NFs). To uncover unusual patterns, AFs focus on individual behaviors, such as account balance, transaction types, and volumes. NFs map an account's network interactions, examining its connectivity and transaction flows with other entities. These features are selected manually without significance testing, pointing towards a tailored but less empirical approach to feature engineering.

Sequential Forward Selection (SFS). SFS begins with an empty feature set and iteratively adds the feature that yields the greatest performance improvement based on a defined criterion [45]. The process continues until no further gains can be achieved by adding additional features.

Recursive Feature Elimination (RFE) vs. Voting-Based. RFE iteratively discards the least significant features based on their impact on model performance, reassessing the reduced feature

TABLE II: A comparison of the feature selection algorithms.

Work	*Feature Group*	*Selection*	*Algorithm*	*Selection Type*	*Interpretation*	*Computation*	*performance*
Chen *et al.* [21]	Transaction	✓	Cascading	Hybrid	◐	◐	→
Chen *et al.* [26]	Transaction	–	–	–	–	–	–
Palaiokrassas *et al.* [37]	Transaction	✓	Correlation	Traditional	●	○	↓
Wen *et al.* [38]	Transaction	✓	FE/RFE	Traditional	●	○	↓
Kabla *et al.* [25]	Transaction	✓	Voting	Traditional	●	○	↓
Li *et al.* [40]	Transaction	✓	TTAGN	Learning	○	●	↑
Wu *et al.* [15]	Transaction	✓	Trans2vec	Learning	○	●	↑
Li *et al.* [9]	Transaction	✓	TGC	Learning	○	●	↑
Lin *et al.* [7]	Transaction	✓	Phish2vec	Learning	○	●	↑
Wen *et al.* [41]	Transaction	✓	NN	Learning	○	●	↑
Zhou *et al.* [42]	Transaction	–	–	–	–	–	–
Chen *et al.* [26]	Behavioral	–	–	–	–	–	–
Li *et al.* [9]	Behavioral	✓	TGC	Learning	○	●	↑
Li *et al.* [40]	Behavioral	✓	TTAGN	Learning	○	●	↑
Wen *et al.* [38]	Behavioral	✓	FE	Traditional	●	○	↓
Zhou *et al.* [42]	Behavioral	–	–	–	–	–	–
Dong *et al.* [43]	Content	✓	NN	Learning	○	●	↑
Wen *et al.* [41]	Content	✓	NN	Learning	○	●	↑

(1) Abbreviations: Feature engineering (FE), neural networks (NN).
(2) Metrics: Interpretability, computation, and performance.
(3) Values: unsatisfied (○), satisfied (●), partially satisfied (◐), better performance (↑), worse performance (↓), average performance (→), and no basis (–).

set at each step to identify the most informative subset [46]. This process helps understand which features are least helpful in predicting phishing and optimizes the efficiency by focusing on a smaller, more impactful set [47]. Introduced by Kabla *et al.* [25], the voting-based feature selection method applies three distinct ranking algorithms to evaluate and select features. This method incorporates a holistic view by assessing each feature's correlation with the outcome, its importance within a specific classifier, and the collective effectiveness of feature sets through a majority vote. Such an approach aims to refine the feature set for phishing scam detection models, emphasizing consensus among different evaluative methods.

Other feature selection techniques include TTANG, TGC, Phish2vec, and neural network-based techniques, which are delegated to appendix C for the lack of space.

2) Analysis and Results: The study of feature selection techniques reveals a lack of a uniform standard across studies, where each work employs its unique method, making it challenging to compare these methods directly. This diversity raises critical questions about whether observed discrepancies in model accuracy stem from the learning techniques or from the features considered, which are often not transparently disclosed. Several feature representation and selection techniques are utilized. Some indirectly limit their explainability by trading the computational features for performance or vice versa, as shown in the last three columns in Table II.

Methods. The feature representation and selection methods can be categorized into three groups, as follows:

Learning-based Techniques. These techniques begin with a representation based on an initial concept of nodes and edges and evolve into the final representation using a learning component, e.g., a neural network. Among these methods thus far, the category includes Trans2vec, TTAGN, TGC, Phish2vec, and GNN-based techniques.

Traditional Techniques. These methods begin with predefined features from transaction, network, or content data and iteratively retain those that most impact detection performance, typically measured via a loss function. This category includes correlation analysis, voting, and feature engineering.

Hybrid Techniques. These methods are a blend of attributes from both learning-based and traditional techniques. Among the techniques we have explored thus far, the cascading method is the only one that fits this description.

Feature Selection Comparison. The *learning-based techniques* provide feature representations and automate feature selection, reducing the need for human intervention. However, they often lack interpretability since feature selection is indirect. These methods involve weighting features in an initial embedding and projecting them into different dimensions through multiple iterations determined by the network architecture. This process depends on the specific network used and may not generalize well to different datasets, leading to a computationally expensive feature extraction process that needs to be repeated for each dataset or fold.

The *traditional techniques* differ in ignoring certain features entirely instead of optimizing them through weight adjustments based on the loss function. While this approach generally performs less than automated learning techniques, it provides more interpretable features. Knowing which features are most influential for detection helps us analyze their manipulability and design defenses, which helps us in our experimental evaluation in section V. Traditional methods are typically computationally lightweight but require human intervention to understand features, select ones that generalize, and evaluate their suitability for different datasets.

Hybrid Techniques blend elements of learning-based methods without fully learning feature representations. Exemplified by the cascading approach, they progressively incorporate features from neighboring entities (e.g., nodes, edges) to construct representations. This enables partial automation while preserving interpretability.

Statistical Analysis. Several studies [26], [42] apply statistical analysis to detection features by computing metrics such as minimum, maximum, and mean values. However, these analyses may offer limited insight into feature importance, particularly when the statistics lie within a narrow range, reducing their discriminative utility.

Summary. Table II shows a summary of the feature selection algorithms utilized in the various schemes of phishing detection, if any, categorized against their type, interpretability, computational complexity, and performance. We note that 4 out of 18 detection schemes did not use any form of feature selection. In contrast, 9 (50%) used learning-based techniques, 4 used traditional feature selection approaches, and only 1 used hybrid approaches. This makes the learning-based techniques the most widely used category despite their clear disadvantages in the security domain in terms of their computational complexity and limited interpretability.

ANSWERING RQ1. Various techniques are employed for feature selection in phishing transaction detection. Among the nine examined, five utilize learning-based methods that often produce uninterpretable features, while three explicitly focus on feature selection and one adopts a hybrid approach. Although no standardized protocol exists, feature selection is commonly integrated. However, only a few methods directly assess feature importance, and even when emphasized, the interpretability remains limited.

B. *Phishing-to-Benign Ratio*

In phishing detection, separating benign from malicious transactions is essential, and the phishing-to-benign ratio (Table III) plays a central role. In this direction, our analysis examines four questions: (1) whether this ratio is altered in ML-based phishing detection, (2) what balancing techniques are used, (3) how these techniques reshape the representation of benign versus phishing transactions, and (4) how such balancing ultimately affects generalization.

1) Balancing Phishing Ratios: Phishing involves malicious attempts to deceive users into disclosing sensitive information, such as private keys or wallet credentials, whereas benign refers to non-malicious transactions or accounts. The phishing-to-benign ratio quantifies the proportion of phishing relative to benign activities. Constructing balanced training sets is often essential due to the underlying mechanics of machine learning algorithms, which require such balance to achieve high accuracy in phishing detection [48].

In real-world applications, biased models can reduce accuracy if the classes are imbalanced, leading to a preference for the overrepresented class [49]. Achieving a balanced ratio enhances the models' adaptation to diverse scenarios, reducing overfitting risks [34]. Consequently, maintaining a balanced representation in training data is critical for the models' accuracy. However, this artificial balancing may not reflect the real scenario where the imbalance is intrinsic.

Next, we explore whether these assumptions are maintained in model training and their impact on generalization.

2) Dataset Balancing Techniques: The approaches outlined in Table III use various techniques to optimize the dataset's composition by balancing, which we review with their pitfalls.

No Balancing. Several studies [26], [42], [43], [15] use the original unbalanced datasets, which creates challenges for learning the minority phishing class as models tend to favor the dominant benign class and fail to recognize phishing reliably. While unbalanced training reflects the real-world deployment setting and preserves the dataset's natural distribution, it also limits the model's ability to learn rare malicious patterns.

Filtering Rules. Data cleaning techniques are applied to eliminate outliers or specific data types, thus improving detection accuracy. Chen *et al.* [21] refined their dataset by filtering out transactions and accounts with significant transaction volumes, substantially lowering their count of phishing addresses. Similarly, Wen *et al.* [38] applied a similar strategy by removing accounts with fewer than four incoming transactions or balances below 5 *ether*. Li *et al.* [40], on the other hand, removed transactions before August 2, 2016, and those from exceptionally active addresses. While aimed at pinpointing phishing activities with greater precision, these strategies come with a notable drawback: the inadvertent removal of legitimate transactions that fall outside standard patterns. This risk could diminish the model's capacity to detect a wider array of phishing behaviors, potentially weakening their effectiveness.

SMOTE. The Synthetic Minority Over-sampling Technique (SMOTE) aims to achieve dataset balance by creating synthetic examples of underrepresented classes, thereby improving predictive model accuracy. Palaiokrassas *et al.* [37] used SMOTE with over 54 million Ethereum transactions, identifying only 81 addresses associated with illicit activities. While balancing phishing and benign instances for training, this approach introduces the risk of artificial patterns. Such deviations could compromise the model's effectiveness on genuine data, impeding generalization.

Over-sampling. Over-sampling, used to balance datasets by increasing the minority classes representation, has been applied with variations across studies. Kabla *et al.* [25] duplicated phishing records to achieve class balance, while Li *et al.* [9] employed upsampling to enhance the presence of phishing samples within a large dataset. Although this approach effectively increases phishing representation, it introduces the risk of model bias toward the overrepresented class, potentially diminishing the model's ability to distinguish between benign and phishing activities due to overfitting.

Sliding Window. The sliding window technique processes data by iteratively analyzing fixed-size subsets as the window moves across the dataset. Lin *et al.* [7] proposed a Statistics-Based Sampling (SBS) method that selects transactions within specific blocks to evenly distribute phishing activities. Wen *et al.* [41] employed a sliding window of size 16 to capture overlapping transaction segments, aiming to balance phishing and benign activities by leveraging temporal patterns. However, this method may overlook certain phishing behaviors, potentially limiting the model's overall effectiveness.

Summary. Altering the phishing-to-benign transaction ratio can significantly affect detection algorithms' performance. Introducing synthetic or duplicated phishing transactions for dataset balancing poses a risk of distorting the feature distribution learned by algorithms. Consequently, models might excel with altered training data but struggle to apply their insights to real, untouched data. A common risk is overfitting, where the model overly attunes to the artificial noise from these manipulations rather than discerning genuine phishing patterns. Therefore, while balancing may boost algorithm performance in training, it risks undermining effectiveness with real-world data the algorithm ultimately faces.

ANSWERING RQ2. Varied methods are used to adjust the phishing-to-benign ratio demonstrating that such manipulations can markedly undermine detection accuracy and real-world relevance. Although synthetic balancing seeks to improve phishing representation, it may introduce bias and overfitting, thus hindering generalization.

3) Balancing Analysis And Result: Section IV-B2 reveals a landscape marked by diversity, where different studies introduce various *ad hoc* approaches, complicating the task of directly comparing the efficacy of these approaches. This raises a critical question: are the observed variances in model performance due to the balancing methods or the nature of the data/algorithms? Our work sets the stage for a detailed examination of how each category of balancing technique influences the phishing detection models.

No Balancing. Four studies did not implement any balancing: Chen *et al.* [26], Wu *et al.* [15], Zhou *et al.* [42], and Dong *et*

TABLE III: Studies categorized by work, year of publication, the number of transactions, number of accounts, original data distribution, method of balancing, modified data distribution, and the corresponding ratios before and after balancing.

Work	Transactions	Accounts	Original Data		Method	Modified Data		Phishing-to-benign Ratio	
			Phishing	Benign		Phishing	Benign	Original	Modified
Chen *et al.* [21]	7,795,044	534,820	1,683	7,793,359	Filter rules	323	534,497	0.0216%	0.0604%
Chen *et al.* [26]	–	2,973,382	1,157	2,972,225	No balancing	1,157	2,972,225	0.0389%	0.0389%
Palaio. *et al.* [37]	54,000,000	550,000	10,000	540,000	SMOTE	81	10,000	1.818%	0.81%
Wen *et al.* [38]	20,667,671	52,380	3,135	49,245	Filter rules	992	4,066	6.366%	22.675%
Kabla *et al.* [25]	–	84,664	5,448	79,216	Over-sampling	38,143	79,216	6.877%	48.15%
Li *et al.* [40]	208,847,461	6,844,050	4,932	6,839,118	Filter rules	4,932	–	0.072%	–
Wu *et al.* [15]	3.8 billion	500 million	1,259	1,259	No balancing	1,259	1,259	100%	100%
Li *et al.* [9]	219,927,673	9,237,535	5,639	9,231,896	Downsampling	5,639	25,000	0.061%	22.556%
Lin *et al.* [7]	22,594,499	4,116,315	5,168	4,111,147	Sliding window	301	4,116,013	0.1257%	0.0073%
Wen *et al.* [41]	739,790	44,709	4,709	40,000	Sliding window	43,125	74,838	11.772%	57.624%
Zhou *et al.* [42]	332,670	3,359	1,659	1,700	No balancing	1,659	1,700	97.59%	97.59%
Dong *et al.* [43]	4,161,444	944,705	1,660	1,700	No balancing	1,660	1,700	97.64%	97.64%

al. [43]. Specifically, Chen *et al.*.[26] exhibited notably low phishing ratios at 0.039%, introducing challenges to the learning models due to significant class imbalances. Conversely, Zhou *et al.* [42] and Dong *et al.* [43] presented higher ratios, at 97.59% and 97.64%, respectively, indicating a different set of challenges for accurate phishing detection, such as overfitting and a decreased ability to identify legitimate communications effectively. These extremes in dataset composition underscore the pivotal role of balanced data in training machine learning models for phishing detection.

Filter Rules. Filtering strategies yielded varied outcomes across studies. Chen *et al.* [21] reduced phishing transactions to 323 and benign ones to 534,497, increasing the phishing-to-benign ratio from 0.0216% to 0.0604%. Wen *et al.* [38] achieved a more drastic shift, reducing phishing transactions to 992 and benign to 4,066, raising the ratio from 6.366% to 22.675%—a 256.19% increase, indicating significant deviation from realistic distributions. In contrast, Li *et al.* [40] preserved the original phishing ratio (0.072%), suggesting minimal or undocumented impact from filtering. These differences underscore how filtering rules can substantially influence dataset composition and, by extension, detection outcomes.

SMOTE. Utilizing SMOTE, oversampling, and upsampling enables the inflation of minority classes to adjust imbalances. Palaiokrassas *et al.* [37] applied SMOTE to refine the phishing ratio to 0.81%, albeit at the risk of introducing synthetic biases. Conversely, Li *et al.* [9] leveraged upsampling for 219.9 million transactions, elevating the phishing detection by increasing its ratio from 0.061% to 22% (36 folds). Similarly, Kabla *et al.* [25] utilized oversampling, boosting the ratio from 6.877% to 48.15% (seven folds).

These strategies emphasize the importance of preprocessing in phishing detection, though they vary in overfitting risk and realism. Compared to direct oversampling, SMOTE provides a more diverse but synthetic augmentation of minority classes.

Impact of Oversampling. Using the existing replicated runs of Kabla *et al.* [25], we compare performance with and without the oversampling step. Removing oversampling in this configuration results in an approximate 18% drop in F1 and a 0.07 reduction in AUC. These differences come directly from the configurations reproduced in our evaluation and highlight how oversampling can substantially inflate performance under extreme imbalance. At the same time, the decline in performance without oversampling aligns with our broader finding that aggressive balancing strategies often reduce generalizability, as examined in §IV-B and §VI.

Sliding Window. The sliding window technique involves analyzing a subset of data over a fixed period, which shifts progressively over the entire dataset, to capture dynamic changes in data characteristics. Using this technique, Wen *et al.* [41] and Lin *et al.* [7] observed divergent outcomes in their ratios after implementing their balancing methods. In Wen *et al.* [41], the phishing-to-benign ratio increased from 11.77% to 57.62% (389% surge in phishing). Lin *et al.* [7] reduced the ratio from 0.126% to 0.0073%; i.e., more than 94% reduction. Such outcomes emphasize the sliding window technique's flexibility, which can significantly diminish or considerably amplify the visibility of phishing.

Summary. Our analysis examines preprocessing techniques across multiple studies, outlined in Table III. Notably, seven out of twelve studies showed significant alterations in phishing ratios following the application of balancing methods, with increases reaching as high as 22 folds. These changes underscore the substantial modifications made to address dataset imbalances, reflecting the adoption of various strategies. Nevertheless, among the studies we surveyed, four studies chose not to alter the datasets, staying faithful to the dataset origin in the real-world deployment scenario.

ANSWERING RQ4. Preprocessing influences detection performance where severe class imbalances–particularly low phishing ratios–hindered effective model training. While filtering and synthetic generation improve dataset composition, they risk overfitting and unrealistic patterns that deviate from real-world conditions. These issues highlight the need to preserve data authenticity and relevance to ensure robust phishing detection across diverse scenarios.

C. Features Robustness

Threat-Model Overview. Our robustness analysis assumes an adversary with the ability to manipulate transaction timing, generate new addresses, and coordinate low-cost evasive behaviors, but without the capability to alter confirmed blocks, modify validated transactions, or influence blockchain consensus. These assumptions reflect realistic attacker actions on Ethereum, where evasion is limited by observable costs such as gas-price-driven delays, transaction-fee overheads,

TABLE IV: An analytical evaluation of robustness reported across prior phishing-detection studies. Following their original characterizations and the analysis in Section IV-C, features are labeled as *prone*, *susceptible*, or *resilient* based on whether they were observed to be easily manipulated, manipulable under specific or constrained conditions, or generally stable due to blockchain-level constraints.

Work	*T*	*A*	*ID*	*OD*	*D*	*IS*	*OS*	*S*	*N*	*BN*	*GF*	*NT*	*ST*	*From*	*To*	*Input*	*TD*	*B*	*Prone*	*Suscept.*	Resi.
Chen *et al.* [21]	○	◐																	1	1	0
Chen *et al.* [26]			◐	◐	◐	◐	◐	◐	○										1	6	0
Palaio. *et al.* [37]										●	◐	○	○						2	1	1
Wen *et al.* [38]	○	◐	◐	◐					○					◐	◐				2	5	0
Kabla *et al.* [25]	○	◐								●				◐	◐	◐			1	4	1
Li *et al.* [40]		◐	◐	◐	◐							○							1	4	0
Wu *et al.* [15]	○	◐																	1	1	0
Li *et al.* [9]	○	◐	◐	◐	◐							○							2	4	0
Lin *et al.* [7]		◐								●	◐								0	2	1
Wen *et al.* [41]	○	◐									◐	○					○	◐	3	3	0
Zhou *et al.* [42]	○	◐	◐	◐							◐								1	4	0
Dong *et al.* [43]		◐	◐	◐								○							1	3	0

(1) Features: The features and abbreviations are listed in Table I.
(2) Metrics: prone (○), susceptible (◐), and resilient (●) to manipulation. The last three columns are the sum of these metrics for each studied scheme.

and the resource requirements of large-scale address creation. This threat model aligns with the manipulations examined in Sections IV-C and V-E and bounds the space of feasible adversarial behaviors considered in our robustness evaluation.

With the insights gathered from various studies showcased in Table IV, we classify features from prior research according to their susceptibility to adversarial manipulation and the complexities involved in their detection [50]. Our analysis revolves around how features fare when evaluated based on their robustness against manipulation. We classify the features used in the literature into three categories: (1) prone to manipulation, (2) resilient to manipulation, or (3) susceptible to manipulation. In the following, we identify those features, contrast the literature based on them, and answer some key research questions along the way. These robustness labels reflect how prior studies and our analysis characterize each feature's manipulability: *prone* features are easily modified by attackers at low cost, *susceptible* features can be manipulated under constrained conditions, and *resilient* features are those for which manipulation is limited or infeasible due to blockchain validation rules or structural constraints.

1) Features Prone to Manipulation: Adversaries can avoid detection by standard security models via transaction features manipulating (e.g., adversarial learning). For instance, altering the timing and volume is simple, but detecting these manipulations is challenging because they can blend with legitimate activity. Based on our evaluation, we identify the following features prone to manipulation: time, neighbors, number of transactions, and transaction direction. In the following, we make the case for why those features are prone to manipulation; i.e., not robust.

Time. Time refers to the transaction's timestamp, indicating when it was confirmed on the Ethereum network. Despite the immutability of blockchain timestamps post-confirmation, attackers can subtly manipulate transaction timings to their advantage [51]. For instance, by strategically choosing when to broadcast transactions, they can dodge detection during peak network activity or take advantage of specific market conditions. Moreover, by adjusting transaction fees, attackers can influence the priority of their transactions, either delaying them to decrease visibility to miners or hastening their confirmation through higher fees [51], [52].

Neighbors. This feature captures the number of unique addresses a wallet interacts with but is highly susceptible to manipulation. Adversaries can inflate the neighbor count by generating fake addresses, engage with reputable accounts to obscure illicit activity, or churn funds among controlled wallets to conceal the source [53], [54].

Number of Transactions. This feature captures the total number of transfers or trades associated with an address over time [55]. Adversaries may exploit this by generating microtransactions or distributing activity across multiple accounts [56], [57], [58], aiming to obscure illicit flows or mimic legitimate activity. Strategically timed transactions complicate detection by evading monitoring [57].

Transaction Direction. This feature refers to whether a transaction for a particular wallet is incoming or outgoing. Understanding the flow of funds is key to identifying potential fraudulent patterns. Attackers may manipulate this feature by address hopping; moving funds through multiple controlled addresses to blur the distinction between incoming and outgoing transactions and transaction cycling [12], where funds are circulated among attacker-controlled addresses in a closed loop. These techniques, aimed at disguising fraudulent activity and efficiently partitioning cryptocurrency networks, challenge detection systems [52], [59].

2) Features Resilient to Manipulation: Features of inherent blockchain characteristics or historical data are difficult to alter without leaving a trace. Due to its decentralized and immutable nature, blockchain is supposed to ensure these features are secure and formidable to attackers [60].

Successful Transactions. This feature refers to validated and confirmed transactions. The decentralized and immutable blockchains make manipulation a formidable challenge. Given the distributed ledger's nature, altering a transaction record would require a consensus from most network participants, a near-impossible feat [61], [62]. Moreover, any attempt to modify a transaction would necessitate recalculating the cryptographic hashes for all subsequent blocks in the chain, which is infeasible [63]. The ledger's transparency further bolsters security by enabling independent verification of transactions, thereby enhancing trust and integrity [64].

Block Number. This feature refers to the chronological order of blocks, starting from the genesis block. The task of altering the block number is exceptionally challenging due to the blockchain's immutability and the required consensus [61]. Moreover, the possibility of manipulating block height, such as through a 51% attack where an entity gains control over the majority of the network's hashing power [50], remains theoretically conceivable but is practically improbable [65], [66]. Thus, blockchain design serves as a defense mechanism, preserving the integrity of block numbers.

3) Features Susceptible to Manipulation: Though susceptible to manipulation, these features are likely to be flagged by detection systems using anomaly analysis to identify suspicious patterns. In the following, we review these features.

Amount. This feature is the transfer volume of a cryptocurrency. Attackers opt for round numbers to mimic benign activity or stay below reporting thresholds. However, genuine transactions typically involve fractional amounts due to natural trading fluctuations and price changes [67]. Advanced techniques can identify these anomalies by contrasting them with historical transaction patterns, seeking out regular transactions at specific intervals or sudden spikes in activity that may indicate automation or scripting [68].

From and To Addresses. The *from* address is a unique identifier for the sender's wallet and is debited the sent amount, including any fees. Similarly, the *to address* identifies the recipient's wallet, crediting it with the received amount. Although these addresses become immutable once a transaction is confirmed, adversaries can manipulate them in several ways [69]; e.g., scatter or aggregate funds using multiple addresses [70]. Utilizing mixers to combine transactions hides the funds' source. Engaging in address hopping disrupts pattern analysis [71]. Given their similarity, the *from* and *to* features are susceptible to similar manipulations.

Account Balance. This feature is the net currency in a wallet, which is the total of all incoming minus outgoing transactions. Adversaries may manipulate account balances by, for example, simulating legitimate activity [72]. This could involve dispersing transactions across several accounts or using wash trading to obscure funds [73]. Moreover, attackers might exploit vulnerabilities in smart contracts to divert funds or hijack accounts to change their balances, camouflaging fraudulent activities within normal patterns [74].

Other susceptible features include strength, gas limit and fee, degree, and smart contract input, which we delegate with discussion to appendix D for the lack of space.

Summary. As shown in Table IV, the analysis reveals a dynamic tension between the manipulability and resilience of transaction features. While some features are easily exploited, others benefit from blockchain's inherent security properties. This contrast underscores the importance of careful feature selection and adaptive detection strategies to address evolving threats. The findings advocate for a nuanced, flexible approach to feature selection to enhance phishing detection and strengthen blockchain security.

ANSWERING RQ5. Features range from highly susceptible to resistant to manipulation. Adversaries can mask fraudulent activities using transaction timing, volume, direction, and number of neighbors, complicating detection. Conversely, successful transactions, block numbers, and block IDs benefit from blockchain's inherent security.

Framework Summary. To make the analytical framework explicit, we summarize its components, inputs, and outputs as they are already used throughout Sections IV and V. The framework begins with *data collection*, which consumes raw Ethereum transaction records and outputs normalized representations suitable for further analysis. *Preprocessing* then converts these records into structured numerical attributes, producing the feature matrices used in subsequent stages. *Feature selection* (SFS for Eth-PSD and RFE for CTD) takes these matrices as input and produces reduced feature subsets that capture the most predictive attributes. *Robustness assessment* consumes these subsets to evaluate which features are prone, susceptible, or resilient to manipulation, enabling a systematic comparison across studies. These stages are sequentially connected: each step consumes the outputs of the previous one and feeds the next. We also make explicit the decision rules already applied throughout the evaluation, such as treating changes in AUC greater than 0.02 as meaningful. These clarifications formalize the existing workflow without altering the underlying methodology.

V. Experimental Evaluation

We explore the logic behind feature selection and modeling optimization (§V-B), the balance between phishing and benign transactions (§V-C), the effect of preprocessing (§V-D), and feature robustness (§V-E), supporting the analyses in §IV.

Comparative Context. To situate our findings, we briefly discuss representative temporal and contrastive GNN-based phishing detectors, namely TTAGN and TGC. These models achieve high accuracy by leveraging temporal dependencies and contrastive embeddings; however, their internal computations are less interpretable and harder to audit. Our focus differs: we prioritize transparency and robustness through interpretable models and explicit feature analysis. Adding this context underscores the trade-off between accuracy and interpretability and clarifies how our approach complements, rather than replaces, these more complex GNN-based detectors.

A. Experimental Setup

We base our experimental evaluation on two datasets from Ethereum's transaction network curated for phishing studies, namely Eth-PSD [25] and CTD [26]. We also use three widely-used algorithms in interpretable literature, DT, KNN, and LightGBM, as described in section III-B. We chose those techniques over alternatives for three reasons: first, they are the most performant according to Table I; second, they are easy to interpret; and third, they are widely used. For feature selection, we use SFS and RFE, two interpretable techniques. We use the parameters from the original works where the datasets were introduced for the initial feature sets and their size. We used the stratified 5-fold cross-validation in all experiments to evaluate the model's performance and report the average. In the following, we review the two datasets and their features.

Eth-PSD Dataset Overview. The Eth-PSD, due to Kabla *et al.* [25], is a comprehensive data collection that provides insights into the mechanics of Ethereum transactions, facilitating the analysis and detection of phishing. Eth-PSD incorporates TxHash, BlockHeight, TimeStamp, From, To, Value, Input, and a category label of the transaction as benign or phishing. Features like TxHash and TimeStamp allow for examining transactions and timing, whereas From, To, and Value highlight the flow and magnitude of transferred funds.

CTD Overview. CTD [26] focuses on transaction analysis for detecting phishing using transaction flow metrics and account balance, and captures the nuanced dynamics of transactions susceptible to phishing. Transactional features, such as in-degree and out-degree, capture the transaction counts, while aggregate transaction volumes are captured through degree and strength metrics: in-strength, out-strength, and total strength. The number of neighbors and the inverse number of transactions further discern phishing activity.

Model Coverage. As discussed earlier, we selected Decision Tree, KNN, and LightGBM because they are widely used, inherently interpretable, and achieve the top performance among the classical models evaluated in Table I. Our goal is to construct a transparent and auditable detection pipeline rather than to maximize accuracy through opaque architectures. For completeness, we also conducted preliminary checks using existing implementations of SVM and a simple CNN. These additional models produced the same qualitative ranking trends observed with DT, KNN, and LightGBM, supporting our claim that the relative feature effects and robustness patterns reported in this work are not model-specific.

Averaging and Stability. All reported results in Section V are averaged over a stratified 5-fold cross-validation. Across the five folds, we observe low variability in the reported metrics, and none of the fold-to-fold fluctuations alter the relative ordering or the qualitative trends discussed in the analysis. These stable averages confirm the consistency of the observations we highlight, including the diminishing returns from additional features (§V-B) and the degradation under extreme imbalance (§V-C).

B. Feature Selection and Model Optimization

1) Feature Selection: We experimentally assess the efficacy of feature selection over Eth-PSD [25] and CTD *et al.* [26] to pinpoint key predictive features in both datasets. **Eth-PSD Feature Selection.** SFS, described in §IV-A1, was implemented to determine the optimal number of features for the detection model, and the results are shown in Figure 2. The performance, evaluated by the negative mean squared error, plateaus after adding the third feature. This trend is consistently observed with stabilized performance from three to seven features, highlighting the diminishing returns of additional features. Thus, we conclude that a leaner model with just three features is optimal, striking a balance between simplicity and effectiveness without compromising the model's performance. This experimental validation of SFS corroborates our analysis in §IV-A, where we discussed the efficacy of feature selection methods in improving model accuracy.

To identify influential features, Pearson correlation coefficients are computed and visualized in Table V, revealing input (0.62), block height (0.43), and time (0.43) as the most correlated with the target label. Notably, block height and time exhibit strong associations with phishing behavior, which often manifests in temporally clustered and execution-pattern bursts.

CTD's Feature Selection Process. All of CTD's features were normalized to the same scale. For feature selection, consistent with the original work, we use RFE, which provided insights into the predictive dynamics underpinning Ethereum phishing detection. In all cases, and as in the original work, the initial set of features was the following: in-degree, out-degree, degree, in-strength, out-strength, strength, neighbors, and inverse number of transactions.

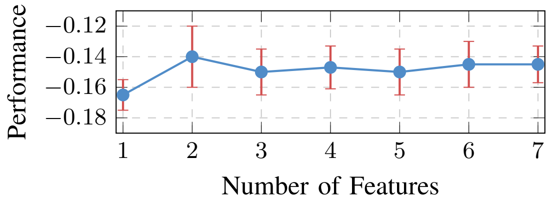


Fig. 2: Performance of the phishing detector (negative mean squared error) as a function of the number of selected features using SFS over the Eth-PSD dataset.

TABLE V: Pearson correlation matrix among transaction attributes. Features used include TxHash (TxH), BlockHeight (BH), and TimeStamp (TS).

Corr.	TxH	BH	TS	From	To	Value	Input	Class
TxH	+1.00	+0.01	+0.01	+0.00	+0.00	-0.00	+0.00	+0.00
BH	-0.01	+1.00	+0.99	+0.35	-0.28	+0.00	+0.26	+0.43
TS	-0.01	+0.99	+1.00	+0.34	-0.27	+0.00	+0.26	+0.43
From	+0.00	+0.35	+0.34	+1.00	-0.28	-0.01	+0.26	+0.36
To	+0.00	-0.28	-0.27	-0.28	+1.00	+0.00	-0.28	-0.27
Value	-0.00	+0.00	+0.00	-0.01	+0.00	+1.00	-0.02	-0.01
Input	+0.00	+0.26	+0.26	+0.26	-0.28	-0.02	+1.00	+0.62
Class	+0.00	+0.43	+0.43	+0.36	-0.27	-0.01	+0.62	+1.00

2) Feature Sets Evaluation: We analyze the sufficiency of feature sets and their combinations. This analysis is grounded in empirical data from performance metrics across diverse feature combinations and uses the work of Kabla *et al.* [25] as an example. The evaluation focuses on the implications of these features on model efficacy, as measured by accuracy, precision, recall, and F1 scores. Table VI compares multiple configurations, incorporating features such as those top-ranked (from, block height, time, and input). These model performance results reveal several insights. This evaluation demonstrates the practical impact of feature selection discussed in §IV-A, where we emphasized the importance of feature selection in enhancing the detection models' effectiveness. ❶ **High Efficacy Combinations.** Both DT and KNN models, employing `from`, block height, time, and input (F, B, T, I), demonstrate superior performance with AUC scores reaching 0.98 and F1 scores reaching 0.96. This blend epitomizes a holistic strategy with multiple dimensions. However, the negligible performance disparity with simpler models supports the need for feature selection. ❷ **Impact of Simplified Sets.** Excluding the input (I) feature still yields high AUC (0.98) and F1 (0.96), questioning its necessity for achieving high precision. This suggests that comparable performance can be attained with a reduced feature set, highlighting the need to justify feature inclusion. ❸ **Minimal Pairs Analysis.** Analyzing minimal feature pairs such as block height and time (B, T) yields an AUC of 0.97, only slightly below that of more complex sets, indicating significant potential for optimizing feature selection.

3) Model Optimization: With a larger set of features, we examine the feature selection as a model optimization problem on CTD *et al.* [26] under a fixed dataset size (40k transactions). Motivated by the findings in the previous section, we look

TABLE VI: Performance DT and KNN models across different feature combinations is evaluated using: (1) Features: From (F), Block Height (B), Time (T), Input (I), and Value (V); and (2) Metrics: Area Under the Curve (AUC), Accuracy (A), Precision (P), Recall (R), and F1 Score.

Combination	KNN					DT				
	AUC	A	P	R	F1	AUC	A	P	R	F1
F, B, T, I	0.97	0.95	0.92	0.95	0.93	0.98	0.97	0.93	1.00	0.96
F, B, T	0.97	0.95	0.92	0.95	0.93	0.98	0.97	0.93	1.00	0.96
F, B	0.96	0.96	0.91	0.97	0.94	0.98	0.97	0.93	1.00	0.96
F, T	0.96	0.96	0.91	0.97	0.94	0.98	0.97	0.92	0.99	0.95
F, I	0.93	0.90	0.87	0.82	0.84	0.97	0.92	0.88	0.87	0.88
F, B, I	0.97	0.94	0.91	0.93	0.92	0.98	0.97	0.92	0.99	0.95
B, T	0.97	0.94	0.91	0.92	0.92	0.97	0.96	0.91	1.00	0.95
B, I	0.97	0.95	0.91	0.93	0.92	0.98	0.97	0.92	1.00	0.96
T, I	0.97	0.95	0.91	0.93	0.92	0.98	0.96	0.90	0.98	0.94
F, B, T, V	0.98	0.97	0.93	0.99	0.96	0.98	0.97	0.93	0.99	0.96
B, T, V	0.97	0.94	0.91	0.92	0.91	0.98	0.97	0.91	0.99	0.95
F, T, V	0.98	0.95	0.91	0.94	0.93	0.98	0.96	0.91	0.98	0.95
T, V	0.97	0.94	0.91	0.92	0.91	0.98	0.95	0.89	0.98	0.93

TABLE VII: Performance vs. feature sets with fixed sample size (40K) from CTD. (1) **Features:** in-degree (ID), out-degree (OD), degree (D), in-strength (IS), out-strength (OS), strength (S), neighbors (N), inverse number of transactions (INT).

Model	**Feature Group**	**AUC**	**Precis.**	**Recall**	**F1**
M1	ID, OD, D, IS, OS, S, N, INT	0.88	0.27	0.09	0.14
M2	OD, IS, OS, S, INT	0.86	0.11	0.03	0.04
M3	IS, OS, S, INT	0.85	0.09	0.03	0.05
M4	OD, D, IS	0.85	0.06	0.02	0.03
M5	D, IS	0.85	0.01	0.01	0.01
M6	S, INT	0.82	0.09	0.02	0.03
M7	IS, OS	0.81	0.01	0.01	0.01

into how far one can push the model simplification while maintaining the performance. We use LightGBM, which previously performed well, as shown in Table I. We start with the comprehensive model and then reduce the complexity by eliminating some features using RFE and Pearson correlation. We make the following observations based on the results shown in Table VII. ❶ **Comprehensive Model.** The comprehensive model (M1) encompassing all the features has resulted in the highest AUC, at 0.88, with modest precision, recall, and F1 scores. ❷ **Large is not Always Better.** M2, a simpler model realized by dropping two features from M1, is only 0.02 worse than the optimal's AUC. This trend, however, does not hold universally since M3 and M4, two models different in size, produced the same performance in terms of AUC. Similarly, M5, M6, and M7, three models of the same size, produced different performances in terms of AUC. Moreover, M6, despite being smaller than M3 and M4, achieved the same precision as M3 (0.09). This highlights the importance of optimizing models to balance performance.

ANSWERING RQ1. Beyond the analytical results in section IV, we show that simplifying the feature set does not affect the model accuracy detrimentally. The nuanced decrease in performance metrics underscores the potential overemphasis on specific features without substantial evidence of impact.

4) Comparison of Models: The algorithms, parameters, and features define the models. To provide insights into the performance of different models, and driven by the insights realized on the importance of feature selection earlier, we evaluate DT and KNN across feature combinations, as shown in Table VI. In the following, we analyze the performance.

TABLE VIII: Model performance for different sample sizes with ratio (P-to-B).

Size	Benign	Phishing	P-to-B	AUC	Precision	Recall	F1 Score
30K	29,897	103	0.0034%	0.88	0.20	0.09	0.12
40K	39,865	135	0.0034%	0.89	0.25	0.10	0.14
50K	49,839	161	0.0032%	0.90	0.20	0.09	0.13
100K	99,723	276	0.0028%	0.90	0.22	0.16	0.18
300K	299,441	559	0.0019%	0.82	0.11	0.15	0.12
500K	499,271	729	0.0015%	0.75	0.12	0.16	0.13

Overall Efficacy. DT and KNN show robust performance across feature combinations. However, DT consistently outperforms KNN in most cases, although marginally.

Feature Combination Impact. Algorithm performance varies with feature set composition. The DT model consistently performs well with specific pairs (e.g., <from, block height>, <from, timestamp>, <block height, timestamp>), sustaining an AUC of 0.98 along with high precision and recall. In contrast, while KNN performs strongly overall, its precision and F1 score decline with reduced feature sets, indicating greater sensitivity to feature selection.

Precision and Recall Balance. The DT model achieves a strong balance between precision and recall when utilizing block height and timestamp features, attaining perfect recall in several cases. This underscores its effectiveness in minimizing false negatives and ensuring comprehensive detection.

ANSWERING RQ3. The performance is influenced by model selection. DT models exhibit a slight advantage overall and particularly excel in precision, while KNN models perform better in terms of recall. These findings underscore that feature selection plays a more critical role than the choice of algorithm in achieving optimal detection performance.

C. Phishing-to-Benign Ratios Under Sampling

This section analyzes how variations in the phishing-to-benign ratio influence model performance and overall effectiveness, drawing on findings from §IV-B and supported by insights from prior literature and empirical studies.

Sampling. In sampling transaction networks, we follow the random walk-based method to extract nodes from the larger CTD dataset as done in [26]. This approach starts from an initial seed node and conducts a walk over one of its neighbors with a probability proportional to the degree. If the next node is not in the list of the visited nodes, it is added, and the process is repeated until the total number of nodes (sample size) is exhausted. Once the sample size is exhausted, the edges between those visited nodes are derived from the original graph. The sample networks are shown in Table VIII. For phishing detection, we use the DT algorithm.

1) Node Size and Ratio Impact: While the absolute number of phishing nodes increases with the dataset size from 103 in the 30K dataset to 729 in the 500K dataset, the proportion of phishing to benign instances significantly drops. Specifically, in the 30K dataset, phishing nodes constituted approximately 0.34% of the data. In the 500K dataset, this proportion dropped to about 0.15%. This shift shows that dataset composition directly impacts model performance, emphasizing the need to balance size with phishing representation for optimal detection accuracy, mirroring the trends observed in §IV-B2

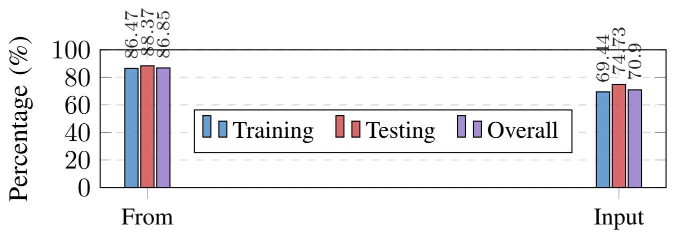


Fig. 3: Comparison of *Input* and *From* features across Training, Testing, and Overall splits. The features exhibit data overlap, reflecting the proportion of matching entries in each category.

where various balancing methods influenced detection efficacy. The decreasing proportion of phishing instances as datasets grow complicates detecting phishing attempts due to a diluted signal-to-noise ratio. This situation underscores the intricate balance required between dataset size and the representativeness of phishing instances for effective phishing detection.

2) *Performance Evaluation:* The phishing-to-benign ratio significantly affects performance across metrics. AUC drops from 0.9 in the 50K and 100K node datasets to 0.75 at 500K nodes, indicating reduced discriminative capability. Precision declines from 0.25 in the 40K dataset to 0.12 at 500K, reflecting diminished accuracy at scale. Meanwhile, recall increases slightly from 0.09 (30K) to 0.16 (500K), suggesting improved identification of phishing instances but at the cost of reduced precision, showing the trade-off between recall and precision as dataset size grows. Considering both precision and recall, the F1 scores across datasets reveal subtle performance variations: 0.12 in the 30K dataset and 0.18 in the 100K dataset, suggesting a balanced improvement. However, this improvement does not persist in larger datasets, with F1 scores reverting to 0.12 and 0.13 for the 300K and 500K datasets.

Takeaway. Adjusting the phishing-to-benign ratio strongly affects performance, but only up to a point–beyond that, models break down. This underscores the need for dataset structures that reflect real-world conditions and for careful balancing to maintain meaningful precision and recall in practical settings.

ANSWERING RQ2. The phishing-to-benign ratio has a substantial impact on detection performance (Table VIII and Section V-C). Empirical results indicate that slight increases in the phishing ratio can enhance accuracy, while greater imbalances lead to performance degradation. This underscores the importance of maintaining a balanced dataset for reliable detection, as further discussed in §IV-B.

D. The Effect of Preprocessing

While preprocessing improves efficiency and initial model accuracy, it raises concerns about its impact on generalization across datasets. This study examines how common preprocessing methods, such as dataset reduction and feature selection, influence a detection algorithm's ability to generalize.

Eth-PSD Dataset Preprocessing. Kabla *et al.*'s preprocessing incorporated *oversampling* to amend class imbalance, resulting in considerable data duplication [25]. Notably, the *Input* feature had a significant overlap of 98,762 instances from a subset of 113,716, contributing to overall data redundancy affecting 33,279 rows. As shown in Figure 3, this high level of duplication weakens the model's ability to distinguish between phishing and benign transactions, increasing the risk of overfitting and making it less reliable.

Examining the *From* and *BlockHeight* attributes reveals substantial redundancy: *From* shows 3,464 overlapping values across 83,209 instances, while *BlockHeight* has 5,519 unique overlaps. This redundancy illustrates the influence of oversampling in promoting a more balanced class distribution (approximately 67.5% to 32.5%) and underscores the trade-offs introduced by such preprocessing interventions.

CTD Preprocessing. CTD's preprocessing, as highlighted earlier, relies on random walks for efficient subgraph sampling and meta-feature engineering to extract node features [26]. This approach, adaptive to dataset size, critically impacts algorithm performance. As highlighted earlier, while this approach does not introduce redundancies, it simply alters the phishing-to-benign ratio, affecting the ability of the models to distinguish between phishing and benign transactions as the sampled dataset complexity increases.

ANSWERING RQ4. Preprocessing datasets influences the generalization capacity. While simplification may improve computational efficiency, it can obscure critical patterns essential for detecting diverse phishing behaviors. Furthermore, although preprocessing can enhance certain performance metrics, it does not reliably translate to improved generalization across varied phishing scenarios.

E. Evaluating Features Robustness

For a thorough analysis, we present initial findings on the impact of feature manipulation on phishing detection performance. While a full evaluation of machine learning robustness is reserved for future work. Building on the insights from (§IV-C), where features were analyzed for their susceptibility to manipulation, we focus on manipulating key features like time, address, input, amount, block number, and successful transactions. These techniques were selected to represent methods adversaries might use to obscure illicit patterns and test model resilience. The goal is to evaluate the resilience of feature sets within transaction datasets when subjected to manipulation, especially those features that are more susceptible to adversarial tactics, like time. This involves assessing how specific feature modifications may obscure patterns, impacting detection accuracy. Understanding these manipulations is essential for identifying vulnerabilities in detection systems and developing strategies to enhance their reliability in detecting fraud-related patterns in real scenarios.

1) *Time Manipulation:* Timestamps play a key role in revealing patterns of illicit activity, yet they are vulnerable to manipulation. As noted in (§IV-C1), time is considered prone to tampering prior to blockchain confirmation. We examine the hypothesis that altering timestamps can obscure behavioral patterns critical to detection, thereby hindering the identification of suspicious activity.

Manipulation Technique. We employed two manipulation techniques. ① *Randomization.* The chronological order was disrupted by randomly shuffling timestamps across the dataset

TABLE IX: Performance Under Manipulation. (1) Features Assessed: Time (T), From (F), Input (I), and Value (V). (2) Evaluation Metrics: AUC, Accuracy (A), Precision (P), Recall (R), and F1 Score. (3) Base Models: M1 and M2 represent baseline models using unaltered features. For manipulation of T, F, and I, we evaluate against M1, which includes F, B, T, and I (1st row in Table VI), reflecting the best-performing model. For V manipulation, we use M2, based on the feature set F, B, T, and V (10th row in Table VI).

Ftr	**KNN**					**DT**				
	AUC	A	P	R	F1	AUC	A	P	R	F1
Manipulated Features										
T	0.78	0.78	0.76	0.50	0.60	0.95	0.95	0.91	0.95	0.93
F	0.97	0.94	0.91	0.92	0.91	0.93	0.92	0.88	0.91	0.89
I	0.83	0.94	0.62	0.32	0.42	0.84	0.92	0.94	0.60	0.58
V	0.83	0.94	0.62	0.32	0.42	0.84	0.92	0.94	0.60	0.58
Baseline Models										
M1	**0.97**	**0.95**	**0.92**	**0.95**	**0.93**	**0.98**	**0.97**	**0.93**	**1.00**	**0.96**
M2	**0.98**	**0.97**	**0.93**	**0.99**	**0.96**	**0.98**	**0.97**	**0.93**	**0.99**	**0.96**

to obfuscate the patterns associated with the timing of transactions. ② *Uniform Distribution.* A uniform time distribution was applied over the specified period (2017–2018) to distribute transactions and mitigate significant concentrations. Both strategies correspond to delaying confirmation, e.g., manipulating fees. These techniques were selected as approximations of tactics attackers might use, such as fee manipulation and delayed confirmations, to mask transaction patterns. We limit our evaluation to modifying a single feature simultaneously to simplify our discussion. **Findings.** Table IX (first row vs. M1) shows that KNN's accuracy dropped from 0.95 (M1) to 0.78, precision from 0.92 to 0.76, and recall from 0.95 to 0.50, resulting in F1 score drop from 0.93 to 0.60, compromising its ability to identify phishing activities accurately.

2) Address Manipulation: To evaluate the effect of address-based evasion, we simulated address hopping in the Ethereum transaction dataset, replicating adversarial strategies aimed at avoiding detection. This aligns with our discussion in § IV-C3, which highlights the ease of manipulating `from` and `to` addresses as a common tactic for evading ML-based detection.

Manipulation Techniques. A custom hashing function was employed, combining the original sender address, a unique salt, and the transaction index to generate distinct addresses, simulating address-hopping used to obscure transactions and evade detection. **Findings.** Table Table IX (second row vs. M1) presents the results, where manipulating this feature causes only a moderate decline. For DT models, AUC drops from 0.97 to 0.92, precision from 0.93 to 0.88, recall from 1.00 to 0.91, and F1 score from 0.96 to 0.89.

3) Input Manipulation: The *Input* feature is pivotal for the functionality of smart contracts [17]. Manipulation, however, is possible due to exploitable defects in the contract's validation protocols [75], [76], [77], which aligns with the earlier findings where input is shown as susceptible to manipulation (§ IV-C3).

Manipulation Techniques. To manipulate the input feature, synthetic adversarial examples were created through input length extension, pattern injection, and randomization of input values. Further details are provided in Appendix E.

Findings. The results in Table IX (third row vs. M1) show that input manipulation significantly impacts KNN performance: AUC dropped from 0.97 to 0.83, accuracy from 0.95 to 0.94, precision from 0.92 to 0.62, and recall from 0.95 to 0.32, leading to a decrease in F1 score from 0.93 to 0.42. DT exhibited a more moderate decline.

4) Amount Manipulation: Manipulation of *value* poses a significant threat, as attackers obfuscate fraudulent transactions by blending them with legitimate ones—aligning with our identification of *amount* as a susceptible feature in § IV-C3.

Manipulation Techniques. ① *Refined amount smoothing.* We adjust the transaction amounts to incorporate randomly generated fractional components. Such a technique is devised to seamlessly integrate suspicious transactions within the flow of legitimate ones, eliminating any noticeable discrepancies. ② *Distributive pattern emulation.* By analyzing the distribution patterns of legitimate transactions and replicating these patterns in manipulated transactions. This strategy aims to create a mask of normalcy. **Findings.** As shown in Table IX (fourth row vs. M2), KNN performance declined notably: AUC dropped from 0.98 to 0.83, accuracy from 0.97 to 0.94, precision from 0.93 to 0.62, and recall sharply from 0.99 to 0.32, resulting in an F1 score decrease from 0.96 to 0.42. The DT model, however, experienced a less pronounced decline.

5) Block Numbers and Successful Transactions Manipulation: Blockchain's design and security mechanisms make it technically infeasible to manipulate features such as the number of successful transactions or block numbers [61]. As discussed in § IV-C2, these features are safeguarded by blockchain's inherent immutability. Although partitioning attacks could potentially compromise these properties, they are costly and rarely feasible in practice.

ANSWERING RQ5. Susceptibility to manipulation varies significantly. Features derived from primary transaction data are particularly vulnerable, leading to notable performance degradation. These results highlight the importance of prioritizing stable features in feature selection and model design for resilience against manipulation.

VI. CONCLUSION

We examined ML-based phishing detection on Ethereum and identified key challenges in feature selection, dataset balancing, robustness, and algorithm choice. Our evaluation framework clarifies how these factors shape model generalization, showing that common practices such as ratio adjustment and oversampling can inflate performance while reducing applicability. We also demonstrated that features such as transaction amount, time, and address are easily manipulated, reinforcing the need for resilient features, careful preprocessing, and algorithms aligned with realistic threat models.

Future Directions and Practical Implications. Practical systems should emphasize harder-to-manipulate features, realistic class distributions, and adversarial validation before deployment. Adversarial strategies will continue to evolve, through timing manipulation, value adjustments, and cross-chain movement, posing challenges for current pipelines. These observations follow directly from the robustness patterns in §IV-C and the evasion behaviors in §V-E. Additional community directions, including evolving benchmarks and structured adversarial testing, are outlined in Appendix F.

References

[1] K. Li, Y. Wang, and Y. Tang, "DETER: Denial of Ethereum Txpool sERvices," in *ACM Conference on Computer and Communications Security, CCS*, 2021, pp. 1645–1667.

[2] H. Heo, S. Woo, T. Yoon, M. S. Kang, and S. Shin, "Partitioning Ethereum without Eclipsing It," in *Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2023.

[3] A. Yaish, G. Stern, and A. Zohar, "Uncle Maker: (Time)Stamping Out The Competition in Ethereum," in *ACM Conference on Computer and Communications Security, CCS*, 2023, pp. 135–149.

[4] C. Schneidewind, I. Grishchenko, M. Scherer, and M. Maffei, "eThor: Practical and Provably Sound Static Analysis of Ethereum Smart Contracts," in *ACM Conference on Computer and Communications Security, CCS*, 2020, pp. 621–640.

[5] P. Li, Y. Xie, X. Xu, J. Zhou, and Q. Xuan, "Phishing Fraud Detection on Ethereum Using Graph Neural Network," in *International Conference on Blockchain and Trustworthy Systems, BlockSys*. Springer, 2022, pp. 362–375.

[6] W. Wang, W. Huang, Z. Meng, Y. Xiong, F. Miao, X. Fang, C. Tu, and R. Ji, "Automated Inference on Financial Security of Ethereum Smart Contracts," in *USENIX Security Symposium*, 2023, pp. 3367–3383.

[7] Z. Lin, X. Xiao, G. Hu, B. Zhang, Q. Liu, and X. Luo, "Phish2vec: A Temporal and Heterogeneous Network Embedding Approach for Detecting Phishing Scams on Ethereum," in *International Conference on Sensing, Communication, and Networking, SECON*. IEEE, 2023, pp. 501–509.

[8] S. Li, R. Wang, H. Wu, S. Zhong, and F. Xu, "SIEGE: Self-Supervised Incremental Deep Graph Learning for Ethereum Phishing Scam Detection," in *International Conference on Multimedia, MM*. ACM, 2023, pp. 8881–8890.

[9] S. Li, G. Gou, C. Liu, G. Xiong, Z. Li, J. Xiao, and X. Xing, "TGC: Transaction Graph Contrast Network for Ethereum Phishing Scam Detection," in *Annual Computer Security Applications Conference, ACSAC*. ACM, 2023, pp. 352–365.

[10] F. Cernera, M. L. Morgia, A. Mei, and F. Sassi, "Token Spammers, Rug Pulls, and Sniper Bots: An Analysis of the Ecosystem of Tokens in Ethereum and in the Binance Smart Chain (BNB)," in *USENIX Security Symposium*, 2023, pp. 3349–3366.

[11] L. Hornuf, P. P. Momtaz, R. J. Nam, and Y. Yuan, "Cybercrime on the ethereum blockchain," 2023.

[12] L. Liu, W. Tsai, M. Z. A. Bhuiyan, H. Peng, and M. Liu, "Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum," *Future Gener. Comput. Syst.*, vol. 128, pp. 158–166, 2022.

[13] Z. Zhang, T. He, K. Chen, B. Zhang, Q. Wang, and L. Yuan, "Phishing Node Detection in Ethereum Transaction Network Using Graph Convolutional Networks," *Applied Sciences*, vol. 13, no. 11, p. 6430, 2023.

[14] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H. Lee, "Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract," *IEEE Access*, vol. 10, pp. 6605–6621, 2022.

[15] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, and Z. Zheng, "Who Are the Phishers? Phishing Scam Detection on Ethereum via Network Embedding," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 2, pp. 1156–1166, 2022.

[16] W. Wang, J. Song, G. Xu, Y. Li, H. Wang, and C. Su, "ContractWard: Automated Vulnerability Detection Models for Ethereum Smart Contracts," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1133–1144, 2021.

[17] Y. Zhou, D. Kumar, S. Bakshi, J. Mason, A. Miller, and M. D. Bailey, "Erays: Reverse Engineering Ethereum's Opaque Smart Contracts," in *USENIX Security Symposium*, 2018, pp. 1371–1385.

[18] M. Vasek and T. Moore, "Analyzing the Bitcoin Ponzi Scheme Ecosystem," in *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 10958. Springer, 2018, pp. 101–112.

[19] S. Bao, Y. Cao, A. Lei, P. M. Asuquo, H. S. Cruickshank, Z. Sun, and M. Huth, "Pseudonym Management Through Blockchain: Cost-Efficient Privacy Preservation on Intelligent Transportation Systems," *IEEE Access*, vol. 7, pp. 80 390–80 403, 2019.

[20] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting Ponzi schemes on Ethereum: Identification, analysis, and impact," *Future Gener. Comput. Syst.*, vol. 102, pp. 259–277, 2020.

[21] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu, "Phishing Scam Detection on Ethereum: Towards Financial Security for Blockchain Ecosystem," in *International Joint Conference on Artificial Intelligence, IJCAI*. ijcai.org, 2020, pp. 4506–4512.

[22] M. Yao, R. Zhang, H. Xu, S.-H. Chou, V. C. Paturi, A. K. Sikder, and B. Saltaformaggio, "Pulling off the mask: Forensic analysis of the deceptive creator wallets behind smart contract fraud," in *Symposium on Security and Privacy, SP*. IEEE, 2024, pp. 209–209.

[23] S. Somraaj, "Hackers Nab $8M$ in Ethereum via Uniswap Phishing Attack," 2022.

[24] X. Zhou, W. Yang, and X. Tian, "Detecting Phishing Accounts on Ethereum Based on Transaction Records and EGAT," *Electronics*, 2023.

[25] A. H. H. Kabla, M. Anbar, S. Manickam, and S. Karuppayah, "Eth-PSD: A Machine Learning-Based Phishing Scam Detection Approach in Ethereum," *IEEE Access*, vol. 10, pp. 118 043–118 057, 2022.

[26] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, and Z. Zheng, "Phishing Scams Detection in Ethereum Transaction Network," *ACM Trans. Internet Techn.*, vol. 21, no. 1, pp. 10:1–10:16, 2021.

[27] J. Wang, P. Chen, X. Xu, J. Wu, M. Shen, Q. Xuan, and X. Yang, "TSGN: Transaction Subgraph Networks Assisting Phishing Detection in Ethereum," *CoRR*, vol. abs/2208.12938, 2022.

[28] Y. Lou, Y. Zhang, and S. Chen, "Ponzi Contracts Detection Based on Improved Convolutional Neural Network," in *International Conference on Services Computing, SCC*. IEEE, 2020, pp. 353–360.

[29] G. Cola, M. Mazza, and M. Tesconi, "From Tweet to Theft: Tracing the Flow of Stolen Cryptocurrency," in *CEUR Workshop*, vol. 3488. CEUR-WS.org, 2023.

[30] M. Bartoletti, S. Lande, A. Loddo, L. Pompianu, and S. Serusi, "Cryptocurrency Scams: Analysis and Perspectives," *IEEE Access*, vol. 9, pp. 148 353–148 373, 2021.

[31] C. F. Torres, R. Camino, and R. State, "Frontrunner Jones and the Raiders of the Dark Forest: An Empirical Study of Frontrunning on the Ethereum Blockchain," in *USENIX Security Symposium*, 2021, pp. 1343–1359.

[32] L. Su, X. Shen, X. Du, X. Liao, X. Wang, L. Xing, and B. Liu, "Evil Under the Sun: Understanding and Discovering Attacks on Ethereum Decentralized Applications," in *USENIX Security Symposium*, 2021, pp. 1307–1324.

[33] C. F. Torres, M. Steichen, and R. State, "The Art of The Scam: Demystifying Honeypots in Ethereum Smart Contracts," in *USENIX Security Symposium*, 2019, pp. 1591–1607.

[34] R. S. Rao and A. R. Pais, "Detection of phishing websites using an efficient feature-based machine learning framework," *Neural Comput. Appl.*, vol. 31, no. 8, pp. 3851–3873, 2019.

[35] N. Abdelhamid, A. Ayesh, and F. A. Thabtah, "Phishing detection based Associative Classification data mining," *Expert Syst. Appl.*, vol. 41, no. 13, pp. 5948–5959, 2014.

[36] A. Oest, P. Zhang, B. Wardman, E. Nunes, J. Burgis, A. Zand, K. Thomas, A. Doupé, and G. Ahn, "Sunrise to Sunset: Analyzing the End-to-end Life Cycle and Effectiveness of Phishing Attacks at Scale," in *USENIX Security Symposium*, 2020, pp. 361–377.

[37] G. Palaiokrassas, S. Scherrers, I. Ofeidis, and L. Tassiulas, "Leveraging Machine Learning for Multichain DeFi Fraud Detection," *CoRR*, vol. abs/2306.07972, 2023.

[38] H. Wen, J. Fang, J. Wu, and Z. Zheng, "Transaction-Based Hidden Strategies against General Phishing Detection Framework on Ethereum," in *International Symposium on Circuits and Systems, ISCAS*. IEEE, 2021, pp. 1–5.

[39] B. He, Y. Chen, Z. Chen, X. Hu, Y. Hu, L. Wu, R. Chang, H. Wang, and Y. Zhou, "TxPhishScope: Towards Detecting and Understanding Transaction-based Phishing on Ethereum," in *ACM Conference on Computer and Communications Security, CCS*, 2023, pp. 120–134.

[40] S. Li, G. Gou, C. Liu, C. Hou, Z. Li, and G. Xiong, "TTAGN: Temporal Transaction Aggregation Graph Network for Ethereum Phishing Scams Detection," in *The Web Conference, WWW*. ACM, 2022, pp. 661–669.

[41] T. Wen, Y. Xiao, A. Wang, and H. Wang, "A novel hybrid feature fusion model for detecting phishing scam on Ethereum using deep neural network," *Expert Syst. Appl.*, vol. 211, p. 118463, 2023.

[42] X. Zhou, W. Yang, and X. Tian, "Detecting Phishing Accounts on Ethereum Based on Transaction Records and EGAT," *Electronics*, vol. 12, no. 4, p. 993, 2023.

[43] J. Dong, Z. Qin, Z. Fang, X. Chen, Z. Huang, H. Guo, R. Liu, C. Turkay, and S. Chen, "Visual Analytics for Phishing Scam Identification in Blockchain Transactions with Multiple Model Comparison," in *International Symposium on Visual Information Communication and Interaction, VINCI*. ACM, 2023, pp. 17:1–17:9.

[44] L. P. Krishnan, I. Vakilinia, S. Reddivari, and S. Ahuja, "Scams and Solutions in Cryptocurrencies - A Survey Analyzing Existing Machine Learning Models," *Inf.*, vol. 14, no. 3, p. 171, 2023.

[45] D. Ververidis and C. Kotropoulos, "sequential forward feature selection with low computational cost," in *European Signal Processing Conference, EUSIPCO*. IEEE, 2005, pp. 1–4.

[46] M. G. Ismail, M. A. E. Ghany, and M. A. Salem, "Enhanced Recursive Feature Elimination for IoT Intrusion Detection Systems," in *International Conference on Microelectronics, ICM*. IEEE, 2022, pp. 193–196.

[47] N. S. M. Nafis and S. Awang, "An Enhanced Hybrid Feature Selection Technique Using Term Frequency-Inverse Document Frequency and Support Vector Machine-Recursive Feature Elimination for Sentiment Classification," *IEEE Access*, vol. 9, pp. 52 177–52 192, 2021.

[48] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine Learning and Deep Learning Methods for Cybersecurity," *IEEE Access*, vol. 6, pp. 35 365–35 381, 2018.

[49] P. Subba Narasimha, B. Arinze, and M. Anandarajan, "The predictive accuracy of artificial neural networks and multiple regression in the case of skewed data: Exploration of some issues," *Expert systems with Applications*, vol. 19, no. 2, pp. 117–123, 2000.

[50] M. Saad, J. Spaulding, L. Njilla, C. A. Kamhoua, S. Shetty, D. Nyang, and D. Mohaisen, "Exploring the Attack Surface of Blockchain: A Comprehensive Survey," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 3, pp. 1977–2008, 2020.

[51] P. Wei, Q. Yuan, and Y. Zheng, "Security of the Blockchain Against Long Delay Attack," in *International Conference on Advances in Cryptology, ASIACRYPT*. Springer, 2018, pp. 250–275.

[52] M. U. Hassan, M. H. Rehmani, and J. Chen, "Anomaly Detection in Blockchain Networks: A Comprehensive Survey," *IEEE Commun. Surv. Tutorials*, vol. 25, no. 1, pp. 289–318, 2023.

[53] B. Lal, R. Agarwal, and S. K. Shukla, "Understanding Money Trails of Suspicious Activities in a cryptocurrency-based Blockchain," *CoRR*, vol. abs/2108.11818, 2021.

[54] M. Saad, V. Cook, L. N. Nguyen, M. T. Thai, and D. Mohaisen, "Exploring Partitioning Attacks on the Bitcoin Network," *IEEE/ACM Trans. Netw.*, vol. 30, no. 1, pp. 202–214, 2022.

[55] M. Bhowmik, T. S. S. Chandana, and B. Rudra, "Comparative study of machine learning algorithms for fraud detection in blockchain," in *International Conference on Computing Methodologies and Communication, ICCMC*, 2021, pp. 539–541.

[56] O. Konashevych and O. Khovayko, "Randpay: The technology for blockchain micropayments and transactions which require recipient's consent," *Comput. Secur.*, vol. 96, p. 101892, 2020.

[57] N. R. Pradhan, A. P. Singh, S. Verma, Kavita, M. Wozniak, J. Shafi, and M. F. Ijaz, "A blockchain based lightweight peer-to-peer energy trading framework for secured high throughput micro-transactions," *Scientific Reports*, vol. 12, no. 1, p. 14523, 2022.

[58] M. Saad, V. Cook, L. N. Nguyen, M. T. Thai, and A. Mohaisen, "Partitioning Attacks on Bitcoin: Colliding Space, Time, and Logic," in *International Conference on Distributed Computing Systems, ICDCS*. IEEE, 2019, pp. 1175–1187.

[59] M. Saad and D. Mohaisen, "Three Birds with One Stone: Efficient Partitioning Attacks on Interdependent Cryptocurrency Networks," in *Symposium on Security and Privacy, SP*. IEEE, 2023, pp. 111–125.

[60] M. Saad, A. Anwar, S. Ravi, and D. Mohaisen, "Revisiting Nakamoto Consensus in Asynchronous Networks: A Comprehensive Analysis of Bitcoin Safety and Chain Quality," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 844–858, 2024.

[61] P. Zhang, D. C. Schmidt, J. White, and A. Dubey, "Chapter Seven - Consensus mechanisms and information security technologies," *Adv. Comput.*, vol. 115, pp. 181–209, 2019.

[62] M. Saad, S. Chen, and D. Mohaisen, "SyncAttack: Double-spending in Bitcoin Without Mining Power," in *ACM Conference on Computer and Communications Security, CCS*, 2021, pp. 1668–1685.

[63] M. Platt and P. McBurney, "Sybil in the Haystack: A Comprehensive Review of Blockchain Consensus Mechanisms in Search of Strong Sybil Attack Resistance," *Algorithms*, vol. 16, no. 1, p. 34, 2023.

[64] C. Zhang, C. Wu, and X. Wang, "Overview of Blockchain Consensus Mechanism," in *International Conference on Big Data Engineering, BDE*. ACM, 2020, pp. 7–12.

[65] F. A. Aponte-Novoa, A. L. S. Orozco, R. Villanueva-Polanco, and P. M. Wightman, "The 51% Attack on Blockchains: A Mining Behavior Study," *IEEE Access*, vol. 9, pp. 140 549–140 564, 2021.

[66] M. Saad, A. Anwar, S. Ravi, and D. Mohaisen, "Revisiting Nakamoto Consensus in Asynchronous Networks: A Comprehensive Analysis of Bitcoin Safety and ChainQuality," in *ACM Conference on Computer and Communications Security, CCS*, 2021, pp. 988–1005.

[67] C. Wronka, "Money laundering through cryptocurrencies-analysis of the phenomenon and appropriate prevention measures," *Journal of Money Laundering Control*, vol. 25, no. 1, pp. 79–94, 2022.

[68] I. Alarab, S. Prakoonwit, and M. I. Nacer, "Comparative Analysis Using Supervised Learning Methods for Anti-Money Laundering in Bitcoin," in *International Conference on Machine Learning Technologies, ICMLT*. ACM, 2020, pp. 11–17.

[69] J. Wu, J. Liu, Y. Zhao, and Z. Zheng, "Analysis of cryptocurrency transactions from a network perspective: An overview," *J. Netw. Comput. Appl.*, vol. 190, p. 103139, 2021.

[70] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies," in *Symposium on Security and Privacy, SP*. IEEE, 2015, pp. 104–121.

[71] E. G. Barrantes, D. H. Ackley, T. S. Palmer, D. Stefanovic, and D. D. Zovi, "Randomized instruction set emulation to disrupt binary code injection attacks," in *ACM Conference on Computer and Communications Security, CCS*, 2003, pp. 281–289.

[72] P. E. Johnson, S. Grazioli, K. Jamal, and R. G. Berryman, "Detecting deception: adversarial problem solving in a low base-rate world," *Cogn. Sci.*, vol. 25, no. 3, pp. 355–392, 2001.

[73] A. Aloosh and J. Li, "Direct evidence of bitcoin wash trading," *Management Science*, 2024.

[74] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart Contract: Attacks and Protections," *IEEE Access*, vol. 8, pp. 24 416–24 427, 2020.

[75] J. Krupp and C. Rossow, "teEther: Gnawing at Ethereum to Automatically Exploit Smart Contracts," in *USENIX Security Symposium*, 2018, pp. 1317–1333.

[76] F. Gritti, N. Ruaro, R. McLaughlin, P. Bose, D. Das, I. Grishchenko, C. Kruegel, and G. Vigna, "Confusum Contractum: Confused Deputy Vulnerabilities in Ethereum Smart Contracts," in *USENIX Security Symposium*, 2023, pp. 1793–1810.

[77] M. Rodler, W. Li, G. O. Karame, and L. Davi, "EVMPatch: Timely and Automated Patching of Ethereum Smart Contracts," in *USENIX Security Symposium*, 2021, pp. 1289–1306.

[78] K. Thomas, D. Y. Huang, D. Y. Wang, E. Bursztein, C. Grier, T. Holt, C. Kruegel, D. McCoy, S. Savage, and G. Vigna, "Framing Dependencies Introduced by Underground Commoditization," in *Annual Workshop on the Economics of Information Security, WEIS*, 2015.

[79] R. Böhme, N. Christin, B. Edelman, and T. Moore, "Bitcoin: Economics, Technology, and Governance," *Journal of Economic Perspectives*, vol. 29, no. 2, 2015.

[80] M. Pacheco, G. A. Oliva, G. K. Rajbahadur, and A. E. Hassan, "Is My Transaction Done Yet? An Empirical Study of Transaction Processing Times in the Ethereum Blockchain Platform," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 3, pp. 59:1–59:46, 2023.

[81] J. Zhao, R. Y. K. Lau, W. Zhang, K. Zhang, X. Chen, and D. Tang, "Extracting and reasoning about implicit behavioral evidences for detecting fraudulent online transactions in e-Commerce," *Decis. Support Syst.*, vol. 86, pp. 109–121, 2016.

[82] J. Leskovec, J. M. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, p. 2, 2007.

[83] M. Eshghie, C. Artho, and D. Gurov, "Dynamic Vulnerability Detection on Smart Contracts Using Machine Learning," in *Evaluation and Assessment in Software Engineering, EASE*. ACM, 2021, pp. 305–312.

[84] K. Martin, M. Rahouti, M. Ayyash, and I. Alsmadi, "Anomaly detection in blockchain using network representation and machine learning," *Secur. Priv.*, vol. 5, no. 2, 2022. [Online]. Available: https://doi.org/10.1002/spy2.192

[85] S. Zhu, W. Li, H. Li, L. Tian, G. Luo, and Z. Cai, "Coin Hopping Attack in Blockchain-Based IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4614–4626, 2019.

[86] F. Zhou, Y. Chen, C. Zhu, L. Jiang, X. Liao, Z. Zhong, X. Chen, Y. Chen, and Y. Zhao, "Visual Analysis of Money Laundering in Cryptocurrency Exchange," *IEEE Trans. Comput. Soc. Syst.*, vol. 11, no. 1, pp. 731–745, 2024.

[87] M. L. Morgia, A. Mei, F. Sassi, and J. Stefa, "Pump and Dumps in the Bitcoin Era: Real Time Detection of Cryptocurrency Market Manipulations," in *International Conference on Computer Communications and Networks, ICCCN*. IEEE, 2020, pp. 1–9.

Appendix

A. Detection Algorithms

Decision Trees (DT). Non-parametric supervised learning method used for classification, where the model learns decision rules from data features to predict target values. Its intuitive, tree-like structure closely resembles human decision-making, making it easy to interpret and useful for explaining models.

K-Nearest Neighbors (KNN). A simple instance-based learning algorithm that delays computation until prediction, classifying new cases based on similarity to stored examples using distance metrics. Its ability to detect outliers makes it effective in identifying phishing patterns during the detection phase.

LightGBM. LightGBM is a fast, scalable gradient boosting framework known for its high accuracy, especially on large datasets. It builds decision trees sequentially, with each tree correcting the errors of the previous one. This makes it well-suited for imbalanced tasks like phishing detection, where fraudulent cases are scarce compared to legitimate ones.

B. Extended Related-Work Summaries

1) Transactional Features: Wu *et al.* [15] extracted time features, including maximum, minimum, and total transaction times, and amount features (transaction amounts) to address issues like data imbalance. Integrating these features with one-class Support Vector Machine (SVM) and the trans2vec algorithm, they demonstrate success in transactional data use for phishing detection. Similarly, Lin *et al.* [7] and Chen *et al.* [26] reinforced the importance of transactional features. Lin *et al.* employed block numbers and gas limits, while Chen *et al.* explored transaction history, stressing the need for detailed transactional analysis in identifying phishing activities.

2) Behavioral Features: Palaiokrassas *et al.* [37] and Wen*et al.* [38] contributed to understanding the behavioral features in Ethereum transactions. Palaiokrassas *et al.* emphasizes the relevance of transaction success rates and interactions with DeFi protocols, suggesting that behavioral patterns in DeFi can be more indicative of phishing than traditional transactions. Wen *et al.* [38] highlights the vulnerabilities in standard machine learning models, emphasizing the need to detect phishing through behavioral analysis.

3) Content Features: Zhou *et al.* [24] provided a prime example on leveraging content features with transaction network graphs over transaction values and gas. The Edge Aggregated Graph Attention Network (EGAT) is used for feature extraction, showing the importance of content-related aspects in phishing detection. In addition, He *et al.* [39] presented a comprehensive approach that intersects with both behavioral and content features, employing techniques like domain-scoring algorithms and website scanners. This integrated methodology demonstrates the interplay between content analysis and behavioral patterns in combating phishing.

4) Extended Comparative Literature Review: The evolution of phishing analysis exemplifies the field's response to increasingly complex threats. This development highlights the research community's dedication to crafting advanced detection methods and maintaining the security of blockchain transactions. Researchers continue to innovate as the threat landscape evolves, proactively addressing and mitigating potential vulnerabilities.

Some features, such as in-degree (ID) and out-degree (OD), appear in multiple groups due to their distinct roles. In transactional features, they quantify direct transaction activities, such as the number of incoming and outgoing transactions. In behavioral features, they reflect interaction trends, measuring engagement levels with other accounts. In content features, they contribute to structural representations, aiding in network graph-based analysis to detect suspicious patterns.

In alignment with these efforts, Table I categorizes key features into transactional, behavioral, and content groups. This categorization provides a comprehensive overview of the varied feature sets employed in recent studies. The table details how features such as time, amount, in-degree, out-degree, and gas fees are analyzed using diverse algorithms like LightGBM, SVM, and neural networks. Performance metrics such as the F1 score and AUC are also reported. This structured overview not only highlights the focus of each study but also emphasizes the complex strategies followed in tackling phishing transactions.

C. Feature Selection Techniques

TTAGN. Li *et al.* [40] present Temporal Transaction Aggregation Graph Network (TTAGN), a method leveraging Temporal Edge Representation to capture Ethereum transaction dynamics. Using Long Short-Term Memory (LSTM), it analyzes temporal interactions between nodes, enhancing edge representations. An Edge2Node module aggregates these representations with attention mechanisms. Additionally, a Structural Enhancement Module equipped with a GCN analyzes the graph topology, thereby enriching node profiles for more comprehensive feature representation.

TGC. Due to Li *et al.* [9], the Transaction Graph Contrast Network (TGC) constructs a node's ego network for comparative pair creation and subgraph training. Through Random Walk with Restart (RWR) sampling, diverse subgraphs are generated for contrastive learning. The feature extraction process is split into node-level, distinguishing a node against its neighbors, and context-level, differentiating the structural patterns of phishing versus normal addresses. The contrastive approach leverages Graph Neural Network (GNN) encoders to unveil distinct transactional and structural patterns.

Phish2vec. Phish2vec [7] is a feature extraction method that models temporal and structural dynamics in transaction networks. It uses a Temporal-based Sequences Generator (TSG), which applies a random walk guided by transaction amount and timing, to capture temporal flow. A Heterogeneous-based Sequences Generator (HSG) further distinguishes between Contract Accounts (CAs) and Externally Owned Accounts (EOAs), modeling their interaction patterns. These sequences are then embedded using word2vec, producing low-dimensional vectors that reflect the roles of each account.

Neural Networks. Wen *et al.* [41] leverage a composite of neural networks for data representation, anchoring on a Back Propagation (BP) neural network for processing transfer and state features. This layer encodes relational patterns into vectors, which are further scrutinized by Fully Convolutional Networks (FCN) and LSTM units to discern transaction features. Dong *et al.* [43] integrates GNNs, with a spotlight on node2vec, alongside feature derivation methods for a nuanced extraction of blockchain transaction data features. This approach, focusing on structural and transactional data, e.g., time and values, aims to distill key indicators of transaction behaviors for enhanced feature selection.

D. Susceptible Features

Strength. This feature, gauging both in-strength and out-strength (total funds entering/leaving a wallet), offers a view of a wallet's transaction volume and direction. Attackers can manipulate these features to conceal illicit actions within ordinary transaction flows [78]. Methods include altering transaction timing to align with user behavior, splitting or merging transactions to mask transfer amounts, or executing circular transactions among controlled accounts [79].

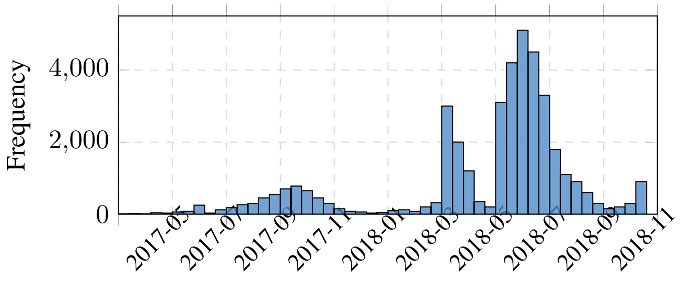
Gas Limit and Fee. On Ethereum, transactions require gas, with the gas limit and fee determining the computational work a transaction can use and its processing speed, respectively [80]. Adversaries may manipulate these features to cause issues like front-running or network congestion. Therefore, gas limits and fees share inherent characteristics; both are vulnerable to manipulation tactics.

Degree. This feature represents the total transaction count associated with an address; in- and out-degree, where the in-degree counts incoming transactions and the out-degree counts outgoing transactions [26]. Adversaries may alter these features, obscuring transaction patterns to bypass detection systems. They might create fictitious transactions to inflate the degree artificially, distribute funds across multiple addresses to elevate the out-degree or employ these tactics in active and complex network engagement [81], [82].

Smart Contract Input. The data fed into a contract when activated, including commands and operational parameters. While typically secure, these inputs can be manipulated due to vulnerabilities in smart contract code. Adversaries might exploit weak validation to alter transaction data, such as through integer overflows [14], causing unintended behaviors. Contracts with poor input validation are particularly at risk, enabling attackers to, for example, redirect funds or perform unauthorized actions. Effective detection relies on robust contract design and stringent validation [83], [16].

E. Impact of Time Manipulation

Figure 4 visually compares the timestamp distributions before and after implementing the randomization technique. Notably, the pre-randomization data displayed pronounced peaks indicative of periods of high-frequency transaction activity. These peaks were significantly reduced post-randomization, leading to a more uniform distribution of timestamps. The manipulation led to several critical findings: ① *Pattern disruption.* The randomization of time successfully masked cyclical daily and weekly transaction patterns, which anomaly detection models typically exploit. ② *Baseline distortion.* Redistributing transactions to achieve a uniform temporal distribution altered the normal baseline of transaction activity. This challenges models that rely on historical data to establish normative patterns. ③ *False negative risk.* A concern with this form of temporal manipulation is the heightened risk of false negatives, where suspicious transactions may evade detection due to their integration into the new uniform timestamp distribution.



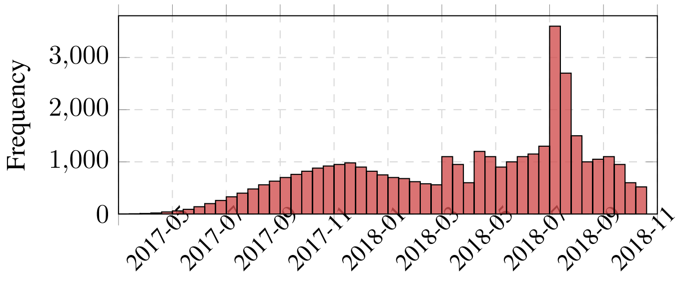


Fig. 4: **Top:** Distribution of timestamps before randomization. **Bottom:** Distribution after randomization. The distribution appears more uniform, yet it still shows variability, with the highest frequency around mid-2018.

D.1. Input Value Exploitation: Adversaries may exploit vulnerabilities within smart contracts, affecting their reliability and security. If not mitigated, these vulnerabilities can lead to substantial financial losses. Two notable vulnerabilities are integer overflow and weak validation checks, which can be utilized to perform input manipulation.

Integer Overflows. Smart contracts written in languages like Solidity often use fixed-size integer types (e.g., uint256). When operations exceed these limits, they can trigger integer overflows–vulnerabilities attackers can exploit by supplying inputs that cause arithmetic wraparounds. This may lead to unintended behaviors such as unauthorized token creation or balance manipulation. For example, in a token contract lacking overflow checks, an attacker could call the transfer function with a large value that, when added to the recipient's balance, causes it to wrap, enabling fund extraction or balance resets beyond the contract's intended logic.

Weak Validation Checks. This vulnerability stems from insufficient input validation, which allows malformed data to be fed into the contract. A contract may fail to verify that inputs fall within expected ranges, conform to the correct type, or satisfy the intended logic. In such cases, attackers can exploit the weak checks to perform unauthorized actions, including withdrawing funds or deploying contracts that interact with legitimate ones in unintended ways. For example, a DeFi lending contract that does not properly validate collateral requirements may allow to obtain a loan with little or no collateral, gaining access to funds beyond the protocol intent.

D.2. Input Manipulation Technique: To assess the vulnerability of detection models to input manipulation, we generated synthetic adversarial examples and examined the model detection accuracy when using them. This process entailed various nuanced tactics, each designed to simulate potential adversarial techniques that could exploit weaknesses within smart contracts. The goal was to gauge the detection models' resilience, or the lack thereof, against input manipulations reflective of real-world attack scenarios. The tactics employed are outlined as follows. ① *Extending input data length.* We devised inputs with lengths that surpass the smart contracts' capacity to handle, aiming to emulate buffer overflow conditions. ② *Pattern injection.* We introduced specific hexadecimal patterns associated with known vulnerabilities into the input data. These patterns were selected based on documented vulnerabilities pertinent to the Ethereum Virtual Machine (EVM) and smart contract bytecode. ③ *Randomizing input values.* Input values were manipulated, randomized, and adjusted across a broad spectrum to induce logical errors or uncover concealed vulnerabilities within the smart contracts. This encompassed modifying numerical values to extremes, randomizing string inputs, and altering transactional data unpredictably. The aim was to reveal logical or computational defects triggered by atypical input values, offering insights into how smart contracts and detection models manage edge cases. We applied these manipulation strategies by selecting a diverse set of base inputs from the Eth-PSD dataset, including pattern appending, input length extension, and value randomization. This approach was intended to test the detection models' current efficacy and to simulate an array of attack scenarios that smart contracts might face in real-world applications.

D.3. Impact of Derived Feature Manipulation: Detection models must leverage the structural properties of networks to identify anomalous patterns and potential security threats within blockchain analytics [84]. These models are heavily reliant on derived features, such as degrees, in-degrees, out-degrees, in-strengths, out-strengths, and transaction frequencies, which are meticulously calculated from transactional data including sender, receiver, time, and amount [15], [7]. This computational foundation underscores the importance of the direct relationship between primary transaction data and the accuracy of derived metrics. Hence, any manipulation of transactional elements can significantly impact the precision of these metrics, challenging the efficacy of anomaly detection models predicated on the interplay between a blockchain's transactional dynamics and its structural characteristics [26].

Specifically, the sender and receiver addresses greatly influence the in-degree and out-degree metrics by quantifying the volume of transactions associated with a particular node. These metrics, reflecting the network's centrality and connectivity, are susceptible to manipulation such as address hopping, which can significantly distort a node's perceived network position [85]. Similarly, the transaction amount is fundamental to calculating in-strength, out-strength, and overall strength, thereby assessing a node's transactional activity. By manipulating transaction values, high-value transfers can be obscured within routine activities, complicating the detection of potentially illicit transactions [86].

Randomizing transaction time can distort frequency patterns, making it harder to detect anomalies based on irregular intervals [87]. Similarly, metrics like comprehensive degree, which combine incoming and outgoing transactions, can be skewed by artificial adjustments [26]. Core attributes such as sender, receiver, timestamp, and amount are foundational to anomaly detection; manipulating them can cascade errors across derived features and compromise detection accuracy. Understanding this dependency is key to building robust models that can detect and resist manipulative behaviors in blockchain-based fraud detection.

As shown in Table I, prior work reveals varied feature dependencies in detection algorithms, highlighting the lack of consistent evaluation of feature robustness. Our findings call for a strategic reassessment of feature selection, emphasizing stable features and advanced analytics to counter evolving adversarial tactics. This holistic approach is crucial for strengthening detection resilience and securing transaction monitoring systems.

F. Extended Discussion from Conclusion

Scope and Generalization. While our evaluation is conducted on Ethereum data, the core insights generalize to other account-based blockchains, such as BNB Chain and Polygon, whose transaction semantics and address behaviors closely mirror Ethereum's. These shared structural properties support the transferability of our findings regarding feature effects and robustness. At the same time, observations tied to Ethereum-specific metadata (e.g., gas-fee structures or block-level pricing dynamics) are ecosystem-dependent and may not apply uniformly across all platforms. We clarify this distinction to bound the scope of generalization without overstating the applicability of Ethereum-specific artifacts.

Model Alternatives. Advanced strategies such as PU-learning, focal or cost-sensitive losses, and new GNN architectures address model-design choices rather than the caveats targeted by our analytical framework. Incorporating such alternatives would redirect this work toward new algorithm development, which lies outside our intended scope. Notably, our empirical findings already demonstrate that interpretable classical models perform comparably to more complex architectures under consistent evaluation settings (see §V).

Broader Research Directions. Although not within the scope of this study, several research directions highlighted by the broader phishing-detection literature remain important for the community. These include standardized, continuously updated benchmarks; adversarial testing and red-team evaluation; and advanced imbalance-handling strategies that reflect evolving blockchain activity. These directions complement, rather than replace, the methodological insights developed in this work.